Backup-and-Restore Scripts

User Guide

Contents:

1. Introduction

This documentation explains the usage of scripts that are part of this backup - and - restore package.

A total of 7 scripts are used for *backup*, *restore* and *data cleanup*:

- **fbackup.sh**: creates a full backup of the home filesystem
- **dbackup.sh**: creates a partial (differential) backup of the home filesystem
- **rmbkdir.sh**: removes all directories created before last full backup
- **bkdircleanup.sh**: cleanup script, removes all unnecessary data from backup directories
- **restoreall.sh**: restores the whole content of the home filesystem
- **restoreuser.sh**: restores the whole content belonging to a specific user
- **restorefile.sh**: restores only the selected items

A configuration (parameters) file is required for correct execution of these programs. More details in the upcoming sections.

Important note:

These scripts work in a coordinated manner. They share specific files (meta-data) and depend on each other in regards of execution. For this reason, it is important to have a good understanding on their functionality. ***Please read the documentation carefully before using this package.***

Disclaimer:

Although thoroughly tested, this script package might still include bugs. It is recommended to do a careful testing prior to using it. Anyone using the package does this at own risk. I will take no responsibility for any issues that might lead to data loss/corruption, nor for any legal matters that might be caused by these issues.

For any questions or bugs/issues reporting please contact me on this e-mail address: ***liviucst@gmail.com***

2. Script environment

2.1. Backup directories

For each backup, a folder (named **full** or **differential backup directory**, depending on script) is created, which contains:

1) **Backed up items**:
    a. user files/directories
    b. miscellaneous items that don't belong to a specific user but are also located in the home filesystem at backup time

2) **Additional files**, which are created and/or maintained by backup script. These are also called **meta-data** and will be referred to in the next section.

All full and differential backup directories are located in a parent backup directory. This will be referred to as **central backup directory** in this documentation.

The central backup folder can either be created by:

a) *user*: when the environment is setup
b) *full backup script*: when running for the first time.

It contains all backup directories and specific meta-data used by scripts.

The central backup directory might even be the mount point of an external hard disk drive. In this case, the HDD would only be used for backup.

Note:

To prevent having a single point of failure, the central backup directory should under no circumstances be created on the same disk where the home partition resides. It should be located on a separate HDD, preferably an external one.

2.2. Configuration file

A **configuration file** (also called **parameters file**) is required for the scripts to run correctly. It is shared by all programs. Each script references it by using the **$param** variable, which contains the absolute path of the file. *Before running the scripts, the user*

*must ensure **$param** is initialized with the correct pathname.*

The configuration file consists of 3 lines, each of them containing a critical parameter:

a) First line: ***absolute path of the central backup directory***

b) Second line: ***absolute path (mount point) of the home partition from where the last backup was made***. If no dedicated home partition was in place, this line will contain the absolute path of the parent directory of the user home folders at last backup.

c) The third line contains the ***absolute path (mount point) of the (actual) home partition***:

- from which items are currently being backed up OR
- to which the items are currently being restored

If no dedicated partition is in place (e.g. home directories are included in the root partition), this line will contain the ***absolute path of the parent directory of the user home folders*** (e.g. /home). The items will either be backed up from or restored to this directory.

Notes:

1) The name and location of the parameters file is at admin's choice. However, it is highly recommended to have it placed in a directory which is not accessible by regular users. A good location would be the /root folder. An alias can be used for having this file quickly edited, e.g. by using vi(m).

2) All scripts except the backup ones (see section 3.1) prompt the user to check the parameters file, change it if required and type 'ok' to resume script execution. This is done for safety reasons, for example to prevent restore of the files in the production home partition in case only a restore simulation needs to be performed. For more details regarding restore simulations see section 4.

3) There are 2 situations when lines 2 and 3 have different content:

*a) **The administrator decides to restore the items to a new (or renamed) directory**. For example, data is accidentally erased from /home. Admin decides to restore the data to /export/home.*

*b) **The administrator decides to change the directory name and resume backups in the new directory (no restore)**. For example /home is changed to /export/home without any impact on user data. Then the backups resume from the renamed directory.*

To ensure operations are performed correctly, both absolute paths (before & after) should be stored in the parameters file. If the mount point/parent directory name remains unchanged (e.g. /home), then same absolute path should be mentioned on both lines.

4) If admin chooses to have the central backup directory created by backup script, it is required that its parent directory exists. The script does not use the -p option with the mkdir command to create the central backup folder!

For example if the chosen central backup directory path is /var/run/media/HDD/BACKUP, it is required that directory /var/run/media/HDD exists prior to having the central backup folder created, otherwise an stop error will be triggered.

2.3. Meta-data

Meta-data are files created and updated by scripts. They are relevant for script operations only and are required for correct execution.

Depending on their purpose, these files are placed either in the central backup directory or in individual backup folders. They are not subject to backup or restore.

Please check APPENDIX B for more details.

***Important note**:*

These files should only be modified by scripts. It is highly recommended to refrain from any manual changes, otherwise errors might occur at script execution.

2.4. Executable script files

As already mentioned, there are 7 executable script files that build this backup & restore package. Their default names are: fbackup.sh, dbackup.sh, rmbkdir.sh, bkdircleanup.sh, restoreall.sh, restoreuser.sh and restorefile.sh. Either of these can be renamed without any consequences on the behavior. The extension is also not mandatory.

The script executables should be contained in a directory which neither belongs to the home filesystem, nor can it be accessed by regular users. A good location would be the **/bin** directory, whose path is usually contained in the environment variable **$PATH**. This means the scripts can be launched from any working directory by just typing their name (and argument if required). Aliases can also be used.

Note:

Please don't forget to update the aliases in case the script names or location change.

2.5. Permissions

All scripts executable files should be owned by root and run in root mode only. There are 2 reasons for this choice. First is to ensure that no errors occur during script execution due to lack of access permissions to specific items. Second is to make sure no regular user has access to the scripts, otherwise the system security would be compromised.

It is recommended to apply following permissions:

a) For script executable files: **-rwx------**
b) For parameters file: **-rw-------**
c) For central backup directory: **-rwx------**

All data created and used by scripts should be owned by root. This includes: *script executable files*, *configuration file* and *central backup directory along with all its content*. The regular users should not have any read/write/execution access to any of them.

3. Scripts

Note:

***it is highly recommended to run the scripts in full screen**. This would usually be available when launching the programs from the no GUI (CLI-only) mode. Not using full screen might cause the output to bounce up & down on screen, especially when a restore or backup directory cleanup is performed.*

3.1 Backup scripts

For each backup script following applies:

1) The script is launched by entering the script name/a path/an alias and pressing ENTER. If no errors occur, no action will be required from user side until end of script execution.
2) No arguments are required.

3) If not all files could be backed up, an error file (**backup_errors.txt**) is created. This file contains the absolute paths of all items that could not be backed up.
4) A detailed log of each backup is written to **operations_log.txt** and **backup_operations_log.txt**.

For details regarding log and error files, see APPENDIX B.

3.1.1 fbackup.sh

This script creates a full backup of the home filesystem. All items are copied to a (full) backup directory, regardless if they changed or not, which means a 1:1 copy of the user home directories is performed.

This is the first script to be run in the environment. A full backup is required for all other scripts to be executed.

Syntax:

[(absolute / relative) PATH]/fbackup.sh

OR

fbackup.sh

OR

[**ALIAS**]

3.1.2. dbackup.sh

This script backs up files and directories that have either been added to home filesystem or changed since last backup. These items are copied to a (differential) backup directory. For each item a full copy is created. The backed up items are also marked as deleted from previous backup directories.

A full backup is required for running this script. After doing the full backup, **dbackup.sh** can be run as many times as required.

The script runs in 3 steps.

Step 1:

The script checks all items that have retained their relative paths to the home filesystem directory since last backup. For each item, one of these possible decisions is taken:

- If item is a directory, it is backed up.
- If item is a file and was modified since last backup, it is also copied to the backup directory.
- If item is a file and was not modified since last backup, then it is ignored.

Step 2:

All items that were added to home filesystem since last backup are copied to the backup folder. This also includes any item which has changed its relative path to home filesystem directory since last backup, e.g. it was moved to another folder. The old relative path is considered a deleted item and is handled in next step.

Step 3:

All items backed up in Step 1 and all items that have been deleted from home filesystem since last backup are marked for deletion from all previous backup folders.

When an item is marked as deleted from a backup directory, it is not actually erased. Instead, its path is removed from the list of backed up items (**log.txt**) and added to a list of "marked for deletion (m.f.d.)" items (**mfd.txt**). Even if not physically removed from backup directory, the m.f.d. instance will be ignored by restore scripts. For more details regarding **log.txt** and **mfd.txt**, see APPENDIX B.

Syntax:

[(absolute / relative) PATH]/dbackup.sh

OR

dbackup.sh

OR

[ALIAS]

Note:

Items that are marked for deletion can be physically erased from backup folders by

*using **bkdircleanup.sh** in order to free disk space. See section 3.3.2 for more details.*

3.2. Restore scripts

For each restore script following applies:

1) Minimum requirement is a full backup.
2) The script is launched by entering the script name/path/alias, an argument (if required) and pressing ENTER. Then the user (admin) is prompted to check parameters file, change it if needed and enter 'ok' to proceed to restore (or any other string to abort script). If no errors occur, no other action will be required until end of script execution, except for **restorefile.sh** (see section 3.2.3).
3) The required items are restored starting with the last full backup and ending with the last created differential backup directory.
4) Following items are ignored:
    a. any backup directory created before last full backup
    b. any m.f.d. item
5) After all items have been recovered, the script applies following permissions:
    a. for directories: **rwxr-xr-x (755)**
    b. for files: **rw-r--r-- (644)**
6) The script does not restore user and group ownership of the items. This should be done by administrator as soon as the program finished execution.
7) If not all items could be restored, a error file (**restore_errors.txt**) is created. It contains the absolute paths of the items that could not be restored.
8) A detailed log of restore operations is created in **operations_log.txt**.

For details regarding log and error files, see APPENDIX B.

3.2.1. restoreall.sh

This script restores the entire home filesystem. No parameters are required.

Syntax:

[(absolute / relative) PATH]/restoreall.sh

OR

restoreall.sh

OR

[ALIAS]

Notes:

1) The path of the home filesystem directory where items are restored could be different from the path of the one from which items had been backed up. For example, the old path might be /home, whereas items are restored to /export/home.

2) Before restoring the items, the script erases the whole content of the given restore directory (except the folder itself). For example, the whole content is erased from /home prior to recovering the items to this directory.

3.2.2. restoreuser.sh

This script restores the home directory of the selected user. The username should be entered as argument ($1) when launching the script.

Syntax:

[(absolute / relative) PATH]/restoreuser.sh [username]

OR

restoreuser.sh [username]

OR

[ALIAS] [username]

Notes:

1) The path of the home filesystem directory where items are restored could be different from the path of the one from which items had been backed up. For example, old path might be /home, while user directory could be restored to /export/home.

2) If the user home folder exists in the home filesystem directory where the restore takes place, it is completely erased along with all its content prior to recovery operation. For example if the restore point is /home and the username is user1, then /home/user1 is erased before restore. Then it is recreated and all user items restored.

3) If no argument is entered, the script will stop. No actions will be performed and the home filesystem content will not be modified whatsoever.

4) The username entered by admin as argument must not contain any slashes on the first or last character position (e.g. don't enter /user1, user1/ or /user1/ as argument). As a

general rule, a username should not include any special characters except underscore (_). For more details, please check the guidelines of your operating system regarding user creation.

5) The user to be restored must have been previously backed up under the same username, otherwise the script will trigger an error and stop. For example, it is not allowed to have the user backed up as user_a and restored as user_b. If renaming is required, this should be done after restore.

6) This script can also be used for restoring an individual directory belonging to a specific user. To achieve this, the admin must enter the relative path name of the directory to home filesystem (starting with the username). For example, if the home filesystem directory is /home and directory /home/user1/Documents needs to be restored, the script argument should be user1/Documents. Please note that when entering the argument, rules from point 4) still apply! Also please make sure no double slashes (e.g. user1//Documents) are accidentally entered within the argument, otherwise a strange script behavior will be triggered (no error but no restore either).

3.2.3. restorefile.sh

This script restores a group of items, which can belong to one or multiple users. These are selected by entering a search keyword as script argument.

Syntax:

[(absolute / relative) PATH]/restorefile.sh [keyword]

OR

restorefile.sh [keyword]

OR

[ALIAS] [keyword]

When the script is launched, the user is prompted to check the parameters file and enter 'ok' to proceed with restore.

The script will then search for items and present the search results on screen. The admin has 3 options:

- **Acknowledge the restore** by entering '***yes***'
- **Not agree with the selected items** by entering '***no***'. In this case, he will be prompted to enter a new search keyword. The search and acknowledge process will be repeated.

- **Abort script** by **entering a different string** or just pressing **ENTER**

After acknowledging the restore, the script will start recovering the requested items. If no error occurs, no other action will be required from admin until script finishes execution. Please note that acknowledging the restore when no items were found leads to script termination.

After restore operations have finished, the script checks if all required items were restored. Based on this check, there are 3 possible scenarios:

1) **No restore errors occurred, all requested items were restored**.

In this case the script ended successfully and following information is displayed: "***All items were successfully restored***". No action required from admin.

2) **Restore errors occurred, not all requested items were restored**.

In this case following information is displayed:

"***Some items could not be restored***
For details please check:
-> /[path]/operations_log
-> /[path]/restore_errors"

This means following issues happened:

a) ***Restore errors.*** These are visible in **restore_errors.txt**. A restore error occurs when an item whose path is contained in the **log.txt** file of a backup folder cannot be restored to the home filesystem directory. The entry contained in **restore_errors.txt** indicates the absolute path of the item as contained in the backup directory from which it should have been restored.

b) ***Some items could not be restored, due to errors that occurred during one of the previous backups***. Otherwise said, some items required for restore were not contained in backup directories, as they hadn't been previously backed up. However their paths are contained in **olddir.txt** (see APPENDIX B for more details), so they were also listed if matching the search keyword. Admin needs to check both **restore_errors.txt** and **operations_log.txt** to determine if this was the case.

The check should be done as follows:

- If the same number of items that could not be restored is listed in both files, then

only restore errors occurred.
- If the number of failed item paths listed in **operations_log.txt** exceeds the one from **restore_errors.txt**, then type b) errors also occurred. These errors are reflected by entries from **operations_log.txt** that don't match the ones from **restore_errors.txt**.

A failed item from ***restore_errors.txt*** matches one from ***operations_log.txt*** when their paths look as follows:
- In *restore_errors.txt*: [**absolute backup directory path**]/[RELATIVE PATH]**/**[*NAME*]
- In *operations_log.txt*: [**absolute home filesystem directory path**]/[SAME RELATIVE PATH]/[*SAME NAME*]

Example:
- In *restore_errors.txt*: **/var/run/media/BACKUP/fbackup_02.11.2014_10:30:35**/user1/Documents/*test*
- In *operations_log.txt*: **/home**/user1/Documents/*test*

3) **No restore errors occurred, however not all requested items were successfully restored**.

In this case, following information is displayed:

"**Some items could not be restored due to errors that occurred at previous backup For details please check: [path]/operations_log**"

No restore errors occurred, however some items could not be restored, as they are not contained in backup directories due to errors at previous backups (see previous scenario for more details).

Notes:

1) The paths displayed on screen as search result after entering the keyword are the absolute paths the items would have after restore.

2) The acknowledge options (yes/no) are case sensitive. They should be entered in small characters, otherwise the third option (script aborted) applies.

3) Before actual restore takes place, the script will search the home partition for existing instances of items to be recovered. This happens after user acknowledged the restore operation. All found items will be erased. Directories are deleted along with the whole content (including items which might not be contained in the restore list). For this

reason, the user should carefully check the list of items to be restored against the actual content of the home filesystem prior to acknowledging the recovery operation, otherwise part of the data could be irreversibly lost.

For example, if /home/Test1/Documents appears in the "to restore" list, the whole directory will be deleted from /home/Test1 prior to restore. If user only requires part of the directory to be restored he should NOT acknowledge the operation.

4) Please run this script in full screen! The program uses a "more" command to display found items. If more items than the maximum number of screen lines are found, then the upper part of the screen will be blank (items are being listed in the lower part). This is a normal behavior, which ensures no items are pushed beyond the upper part of the screen before the user can check them. While scrolling through the items, the blank part is progressively being filled. After all items have been listed, a text is displayed, which prompts the user to acknowledge the restore operation.

5) If using ENTER to list the item paths line by line, please pay attention not to press it after all items have been shown on screen. If pressing it again after the prompt is displayed, the script will be aborted. For a large number of items the SPACE key would be a better listing option (one screen at a time is displayed).

6) Please note that the search keyword is fed to a regular "grep" command. The command has no options. You may include any special characters permitted by normal grep (like ^ for beginning of line, $ for end of line, etc). No extended grep (egrep) is currently included by script.

3.3. Cleanup scripts

For each script following applies:

1) The script is launched by entering the script name/path/alias and pressing ENTER. Then the user (admin) is prompted to check the parameters file, change it if required and enter 'ok' to proceed with cleanup (or any other key combination/ENTER to abort). If no error occurs, no other action will be required until script finishes execution.

2) No arguments are required.

3) These scripts are used for freeing disk space, especially if large files are being backed up frequently.

3.3.1. rmbkdir.sh

This script erases all backup directories created before last full backup.

It will only perform an erase if there are at least 2 full backup folders in the central backup directory.

The operations performed by script are not logged to any file.

For details regarding possible errors see section 5.2.1.

For details regarding used file see APPENDIX B.

Syntax:

[(absolute / relative) PATH]/rmbkdir.sh

OR

rmbkdir.sh

OR

[**ALIAS**]

3.3.2. bkdircleanup.sh

This script attempts to erase items which are marked for deletion from active backup directories. Minimum requirement is a full and a differential backup.

Syntax:

[(absolute / relative) PATH]/bkdircleanup.sh

OR

bkdircleanup.sh

OR

[**ALIAS**]

There are 4 types of m.f.d. items:

a) **Files**. These are always physically erased by script.

b) **Empty directories**. These could have either been empty before the script was run or contained only m.f.d. items, which were deleted beforehand. The script erases them too.
c) **Directories which are not empty**. These will not be deleted, as they contain items which were not marked for deletion.
d) **Items not found by script**. No action is performed.

All physically removed items have their paths moved from **mfd.txt** to **deleted.txt**. The items not found by script have their paths written to **notfound.txt**. Non-empty directories are kept in **mfd.txt**. If no such directories are found during operation, **mfd.txt** will be erased after the backup directory has been cleaned up. For more details see APPENDIX B.

The script operations are logged to **operations_log.txt**.

Note:

*Active backup directories are the ones created starting with last full backup (including the full backup folder). Directories created before last full backup are ignored by backup & restore scripts and by **bkdircleanup.sh**. They can be erased by calling **rmbkdir.sh**.*

4. Environment setup and script execution

Following steps need to be performed when using the script package.

Step 1: copy the scripts into the chosen directory (see section 2.4).

Step 2: create the parameters file in the chosen location (see section 2.2).

Step 3: open each script and update the **$param** variable with the absolute path of the parameters file (see section 2.2).

Step 4: change permissions of the following files (see section 2.5):
- scripts: **-rwx------**
- parameters file: **-rw-------**

Step 5 (optional): create the *central backup directory*. Change its permissions to **drwx------** (see section 2.5).

Step 6: update the first 3 lines of the parameters file (see section 2.2).

Line 1: absolute path of the central backup directory

Line 2: absolute path of the home filesystem directory from which items are being backed up
Line 3: same as line 2, can be changed later if required (see section 2.2)

The format of the absolute path should be: /[**dir1**]/[**dir2**]/**.../**[**directory name**] for each line. For example: for home directory it should be /home, not /home/ (or any other combination of '/'). Not following this guideline might lead to incorrect scripts execution.

Please check that the absolute paths written to the configuration file exist (for lines 2 and 3) and are correct (for all lines).

After all 3 lines have been updated, save and close the file.

Step 7 (optional): create a friendly script execution environment.

This means:

a) **Create aliases for certain execution models**:

- For each script
- For different combinations: differential backup and shutdown, differential backup and restart, etc.
- For parameters file editing

b) **Update the crontab file or any other relevant files for automatic/scheduled script execution if required**.

For aliases suggestions see APPENDIX A.

Step 8: run *fbackup.sh* to perform a full backup.

Step 9: run *dbackup.sh* as often as required.

Important note:

If admin decides to change the mount point of the partition containing the user home directories, following steps are required from a backup point of view:

a) *Highly recommended (but not mandatory):* ***run a full backup prior to doing this change***.
b) *After mount point has been modified, the* ***parameters file should be changed*** *as*

follows:

- ***the second line should contain the previous home filesystem directory path*** *(so this line should remain unchanged), e.g. /home.*
- ***the third line should contain the new home filesystem directory path****, e.g. /export/home*

c) ***Run a new (full/differential) backup when needed***. *After the backup has finished, the script will change the value of the second line to that of the third line. This is required so next backups run correctly.*

d) ***If a restore is required right after changing the mount point and no backup has been done yet (e.g. a user accidentally erases items), the settings at point b) apply***. ***After the restore script has finished execution, these settings should not be modified prior to running next backup (the backup script will take care of this)***. *Please note that only the backup scripts can modify the configuration file (see point c)).*

Step 10 (optional): after first differential backup has been performed, use *bkdircleanup.sh* to free disk space.

Step 11 (optional): after doing a new full backup, use *rmbkdir.sh* to remove previous backup folders.

Step 12 (optional): periodically a ***restore simulation*** can (and is recommended to) be carried out. For this purpose, a location should be chosen for restore. The path of this location would be entered on the third row of the parameters file. The files contained in the currently existing backup folders would then be recovered to this restore point.

Notes to this step:

1) As this is only a restore simulation (and not an actual restore) ***the user/admin should make sure that the chosen restore location is different from the actual home partition!*** *It should also not reside on the same physical HDD with the home fileystem, otherwise it would consume disk bandwidth and therefore slow down user operations. Instead, it could be created on the same disk where the backup directories are placed or most recommended on a completely separate internal/external HDD.*

2) It is also highly recommended to ***create new instances of the restore scripts****, which would be used for restore simulation only. For example, copy restoreall.sh to restoreall_test.sh. Also,* ***a separate parameters file should be used****. This would decrease the number of times the parameters file needs to be modified prior to running a script and would also reduce the probability of human errors.*

3) ***If any errors are noticed when running the restore simulation, a new full backup of***

the home filesystem should be carried out. *The existing backup directories should then be erased. After doing the full backup, a new restore simulation should be carried out to check if errors persist. If this still happens, then most probably some hardware errors (faulty HDD) could be the cause, so the HDD should be replaced. Then new backup and/or restore operations (depending on which HDD (source/destination) caused the issue) should be initiated.*

5. Troubleshooting

This section presents errors that might occur during script execution. See below table for details.

Notes:

1) ***Some of these errors can be avoided by not performing manual changes to the central backup directory and any included items.*** *Manual changes means any modifications done by user/admin to the content of the central backup directory, other than the ones performed by scripts. Such modifications should NOT be performed, unless there is a good reason. Manual changes can lead to serious issues including data loss and corruption.*
2) ***This script does not provide advanced error checking techniques, like checksums.*** *This means some errors, like the ones caused by faulty hard disks might remain undetected. Although these errors are rare, this is something to be aware of. Possible measures to be taken to mitigate these issues might be:*
   - ***running regular restore simulations***
   - ***doing alternative backups on several HDDs****. For example use 2 identical hard disks for backup. Each week do one full backup one one of the disks, while differential backups could be done once a day on every disk.*
3) ***It is recommended to do a health check of the involved HDDs prior to performing a full/differential backup.***
4) ***If a power outage occurs during script operations, it should be assumed that data corruption occurred****. Please run a health check of the involved HDD. Then following actions should be taken:*
   - *If outage occurred during a backup or when using **bkdircleanup.sh**, please run a new full backup. Previous backup directories should be deleted.*
   - *If outage occurred during a restore, please re-run the restore script.*
   - *If outage occurred during rmbkdir.sh operation, please check if directories created since last backup (including the full backup folder) have been affected. If yes, please run a full backup and erase all previous backup directories using **rmbkdir.sh** or manually. If not, please re-run the backup directory erase script if possible or erase all directories not contained in **backuplist.txt** manually.*

*When manually erasing directories, please pay attention not to delete a directory contained in the **backuplist.txt** file. If this occurs, a full backup is required.*

5.1. Exit codes

Following exit codes are used by scripts, depending on how they get terminated:

- **0**: *normal termination* (script successfully executed)
- **1**: *system errors* (e.g. a required file, like **backuplist.txt** is not found)
- **2**: *aborted by user* (e.g. when user is prompted to check the parameters file and type 'ok', he chooses a different key combination, which terminates script)
- **3**: *missing argument*. This happens when the required argument is not entered (username not entered for **restoreuser.sh**, no search keyword entered for **restorefile.sh** – see sections 3.2.2 and 3.2.3 for details).
- **4**: *invalid argument* (e.g. administrator attempts to restore a user that hasn't been previously backed up – see section 3.2.2 for details)
- **5**: *no action required from script* (e.g. when using restorefile.sh, user acknowledges the restore even if no paths were retrieved when using the search keyword)

5.2. Error types

5.2.1. Script abort errors

The below table presents the types of errors that lead to script termination. Admin can identify each error based on the error message generated by script and use this info to retrieve correct information from table and thereby troubleshoot the script execution.

Nr	Error message	What it means	Affected scripts	Exit code	Troubleshooting
1	***Invalid home filesystem path: [PATH] Script aborted***	Either the directory containing the user home directories does not exist or the absolute path entered in the parameters file is missing or incorrect.	**fbackup.sh dbackup.sh**	1	1) Check if the home partition exists and is correctly mounted into the filesystem hierarchy. If user home directories are part of a larger partition (e.g. included in the root partition), check if their parent directory (e.g. /home) exists and is correctly recognized by system. Note the absolute path of the mount point/parent

		For example characters might be missing or a relative path (instead of an absolute one as required) was entered.			directory. 2) If all the above apply, please check the third line of the parameters file against the absolute path of the directory/mount point. They should match 1:1. Correct the configuration file if required and then run the script again.
2	***Cannot create central backup directory: [PATH] Script aborted***	The full backup script cannot create the central backup directory on the designated partition/hard disk. This folder should either exist prior to running the script or be created by **fbackup.sh**. However if none of these apply, this error is triggered.	**fbackup.sh**	1	1) Check if the hard disk is physically connected to the hardware system and is correctly recognized by OS. 2) Check if the partition designated to include the central backup folder is correctly mounted on the system. Note the mount point and the absolute path that the central backup directory should have (might coincide with the mount point if the whole partition is used for backup). 3) If all the above apply, please check if the path contained on line 1 of the parameters file matches the central backup directory path determined at previous point. Correct the configuration file if required and then run the script again.
3	***Central backup directory does not exist or the path <PATH> is invalid Script aborted***	The central backup directory does not exist on the hard disk designated for backup or its absolute path is not correctly entered in the parameters file.	**dbackup.sh** **rmbkdir.sh** **bkdircleanup.sh** **restoreall.sh** **restoreuser.sh** **restorefile.sh**	1	1) Check if the hard disk is physically connected to the hardware system and is correctly recognized by OS. 2) Check if the partition designated to include the central backup folder is correctly mounted on the system. Note the absolute path of the mount point. 3) Check if the central backup directory exists on the

					designated partition (use ls command). Note the absolute path of the directory. 4) If all the above apply, please check if the path contained on line 1 of the parameters file is correct and matches the absolute central backup directory path 1:1. Correct the configuration file if required and run the script again.
4	***File [/…/backuplist] not found Script aborted***	File ***backuplist.txt*** does not exist, probably because it has been deleted, moved or renamed.	**dbackup.sh bkdircleanup.sh restoreall.sh restoreuser.sh restorefile.sh**	1	cd to central backup directory and check if the file exists and is correctly named. The name should be **backuplist** (no extension) and the parent directory should be the central backup folder. Please correct these issues (if possible) and run the script again.
5	***File [/…/olddir] not found Script aborted***	File ***olddir.txt*** does not exist, probably because it has been deleted, moved or renamed.	**dbackup.sh restorefile.sh**	1	cd to central backup directory and check if the file exists and is correctly named. The name should be **olddir** (no extension) and the parent directory should be the central backup folder. Please correct these issues (if possible) and run the script again.
6	***No full backup directory found Script aborted***	Either the full backup directory had been renamed/moved /erased or the first line of ***backuplist.txt*** does not reflect its name correctly.	**dbackup.sh bkdircleanup.sh restoreall.sh restoreuser.sh restorefile.sh**	1	1) cd to central backup folder and check if the full backup directory exists. 2) If the folder exists, please check if it is correctly named. The name of the full backup directory should have format *fbackup_[day.month.year_hour :min:second]*. 3) If all the above apply, please check if the directory name is correctly reflected on the first line of **backuplist.txt**. The line should only contain the exact name of the folder (no absolute path).

					4) After all these issues have been solved (if possible) please run the script again.
7	***Maximum one full backup has been performed*** ***No backup directories to be erased*** ***Script aborted***	Only one full backup has been performed since environment setup. For this reason ***old_backuplist.txt*** does not exist. Consequently no backup directory can be removed.	**rmbkdir.sh**	5	cd to central backup directory. Check if **old_backuplist.txt** had been accidentally erased/moved/renamed. This can be done by comparing the number of backup folders to the number of entries from **backuplist.txt**. If there are more folders than entries, then one of the above applies to **old_backuplist.txt**. If not, run the script again after the next full backup has been performed.
8	***All backup directories created before last full backup have already been removed*** ***No backup directories to be erased*** ***Script aborted***	Script has already been run after last full backup was done. Consequently ***old_backuplist.txt*** is empty.	**rmbkdir.sh**	5	If **old_backuplist.txt** had not been manually emptied (check number of backup directories against number of entries from **backuplist.txt** as mentioned for previous error), then the script should be run again after next full backup has been performed.
9	***The home filesystem directory used for restore does not exist or path [PATH] is invalid*** ***Script aborted***	Either the home filesystem directory where the items should be restored does not exist, or the absolute path entered in the parameters file on the third line is incorrect. For example characters might be missing or a relative path (instead of an absolute one as required) was	**restoreall.sh** **restoreuser.sh** **restorefile.sh**	1	1) Check if the home partition where items should be restored exists and is correctly mounted in the filesystem hierarchy. Note the absolute path of the mount point. If user home directories should be part of a larger partition (e.g. included in the root partition), check if their parent directory (e.g. /home) exists and is correctly recognized by system (use ls command). 2) If all the above apply, please check third line of the parameters file against the absolute path of the parent directory/mount point. They should match 1:1. Correct the

		entered.			configuration file if required and then run the script again.
10	***Cannot reach current backup directory [PATH] Script aborted***	The current differential backup directory might have been accidentally erased/moved/renamed or the current line of ***backuplist.txt*** does not reflect its name correctly.	**dbackup.sh** **restoreall.sh** **restoreuser.sh** **restorefile.sh** **bkdircleanup.sh**	1	If this error occurred when running **bkdircleanup.sh** or **dbackup.sh**, please run a new full backup. If error was encountered during a restore, please follow the below steps: 1) cd to central backup folder and check if the backup directory exists. 2) If the folder exists, please check if it is correctly named. The name should have format *dbackup_[day.month.year_hour:min:second]*. 3) Please repeat this check for all other differential backup folders. 4) Please check if the name of each differential backup folder is correctly reflected in **backuplist.txt**. Make sure all names are entered one per line, in the same order in which the directories were created. The first line should be the full backup directory name. 5) After all these issues have been solved (if possible) please run the restore script again. If issues could not be solved, then a manual restore should be attempted.
11	***Cannot find [.../log] Script aborted***	The ***log.txt*** file of the current backup directory does not exist. It might have been	**dbackup.sh** **restoreall.sh** **restoreuser.sh** **restorefile.sh**	1	1) For **dbackup.sh** If this error occurred, you should run a full backup and remove all previous backup directories using the **rmbkdir.sh** utility.

		accidentally deleted, renamed or moved somewhere else.			2) For restore scripts: Check current backup directory. If **log.txt** does not exist, you should manually re-create it as follows: ***cd [backup directory];*** ***find . | sed '1d' | cut -c 2- | sort > log*** Both commands should be run in root mode. After re-creating the file, please run the restore script again.
12	***No username entered*** ***Script aborted***	No username was entered as argument when running the user restore script.	**restoreuser.sh**	3	Run the script again by entering the username argument along with the script name/path/alias.
13	***Invalid username [USERNAME] or no previous backup done for this user*** ***Script aborted***	Possible reasons are: a) Entered argument contains the slash (/) character on first or last position (or both) b) User name does not match any previously backed up user, meaning no user home directory with this name exists in any of the backup folders. c) Other	**restoreuser.sh**	4	1) Check backup directories to determine if the user home directory that must be restored exists. 2) If the answer is yes, then the script argument might have been entered incorrectly. Run the script again by entering the argument so that it matches the backed up user home directory name 1:1.

		username typo (e.g. one extra letter/special character etc)			
14	***No search keyword entered Script aborted***	No search keyword was entered as script argument.	**restorefile.sh**	3	Run the script again by entering an argument to be used as search keyword.
15	***No items to be restored Script aborted***	Admin acknowledged the restore operation although no items were found.	**restorefile.sh**	5	Run the script again by ensuring minimum one item is found based on the search keyword prior to acknowledging the restore.
16	***Script aborted***	Following 2 scenarios might apply: a) When being prompted for parameters file check, user aborts by entering a keyword different from 'ok'. b) User aborts the item restore script (**restorefile.sh**) by entering a keyword different from 'yes' or 'no' when being asked to acknowledge the restore. Please note that in both cases the keywords are case sensitive (small	**rmbkdir.sh** **bkdircleanup.sh** **restoreall.sh** **restoreuser.sh** **restorefile.sh**	2	No troubleshooting required as it is voluntarily aborted by user.

		letters required).			
17	***Failed scanning [PATH]***	Searching the items contained in the home filesystem directory failed.	**fbackup.sh dbackup.sh**	1	A hard disk error might have occurred or the link between HDD and computer might be faulty (e.g. cable is not properly connected). Please ensure there is a good hardware connection between the devices and try again. If the error still persists there might be a hardware issue.

5.2.2. Non-fatal errors

These are errors that don't lead to script termination. For this reason, no exit code is required. However they should not be ignored. The final screen of each script shows a summary, which also includes the number of occurred errors. This summary is also included in the operation log files. If the number of errors is higher than 0, the admin should check them by using the appropriate error file. Corrective actions might be required.

5.2.2.1. Backup errors

Affected scripts are:

- ***fbackup.sh***
- ***dbackup.sh***

The issues are visible in **backup_errors.txt**. This file is included in each backup directory if errors occurred during backup, otherwise no error file is created. Each entry is the absolute path of the item as contained in the home filesystem at backup time.

The issues should be investigated on a “per-item” basis. If applicable, a manual backup could be done by using the appropriate commands to copy the item to the backup directory. Care should be taken to update **log.txt** accordingly and ensure the item paths remain correctly sorted (in ascending order) in this file. This procedure should only be applied if a backup of this item is absolutely necessary, otherwise it should be avoided as it can lead to errors. A much better option would be to use the **touch** command for updating the change date of the item, so it is backed up by next differential script session.

5.2.2.2. Restore errors

Affected scripts are:

- ***restoreall.sh***
- ***restoreuser.sh***
- ***restorefile.sh***

These errors are visible in the **restore_errors.txt** file, which is included in the central backup directory. No error file will be available if no restore issues occurred. Each entry contains the absolute path of the failed item, as originally included in the backup directory.

These issues should be investigated on a "per-item" basis, by identifying in which backup directory each one occurred. A manual restore might be attempted if necessary.

Regarding **restorefile.sh** some additional checks should be done. For more information see section 3.2.3.

6. Miscellaneous

6.1. Identifying a successful script execution

Following hints could be used to determine if the script execution succeeded or failed:

1) Check first line of **operations_log.txt**, which contains the script execution start date. If this information is not correct (date is older than the one when the script operations began), the script execution didn't finish successfully.

2) The existence of a **temp_operations_log.txt** file in the central backup folder is also a hint that the script execution failed.

3) For backups only:

a) Check **backuplist.txt**. The last line should contain the name of the current backup directory. If not contained, it means the script execution was not successful.

b) The content of **bdate.txt** should be identical to the first line of **operations_log.txt**. The update of **bdate.txt** is the last step performed by backup scripts, so this can be seen as a good confirmation that the script successfully ended. If not identical, then the script did not succeed.

4) As a general rule, as soon as script operations are complete, please check that the required meta-data files exist and are located properly. A detailed description of these can be found in APPENDIX B, however a short summary of what needs to be checked can be provided here:

- **operations_log.txt** should be located in the central backup directory. This file is updated by all scripts except **rmbkdir.sh**
- the **backup_operations_log.txt** should be located in each backup directory. It is dedicated only to a specific backup.
- the **log.txt** file should be located in each backup directory. It contains the paths of all backed up items except the ones which were marked for deletion by subsequent backups. These ones are contained in **mfd.txt**, which is located in the same backup folder. If all m.f.d. items had been deleted by ***bkdircleanup.sh***, this file is removed. For details see section 3.3.2.
- **deleted.txt** only exists in backup directories where items had been removed by ***bkdircleanup.sh***. It contains the paths of the physically erased items.
- **notfound.txt** only exists in backup directories where items were not found by ***bkdircleanup.sh*** and consequently no action could be performed.
- **backup_errors.txt** is located in each backup folder, where errors occurred during backup
- **restore_errors.txt** is located in the central backup directory. It only exists, if errors occurred at last restore operation.
- **erase_errors.txt** is generated by ***rmbkdir.txt***, if minimum one required backup directory could not be erased. It is located in central backup directory.

If additional meta-data files (other than the above mentioned) are contained in the central backup directory or the backup folders, then the script execution didn't finish successfully. Before performing a new script operation, it is strongly recommended to erase these extra files manually. They can be identified in APPENDIX B by having value Yes in the last column (Removed after script execution). Not erasing these items might cause errors/unexpected behavior when re-running the programs.

6.2. Script limitations

Following limitations apply to these scripts:

1) ***They are designed for disk-to-disk file backup***. No tape backup should be performed as moving backup directories to tape will prevent the 3rd step of the differential backup script (namely marking items for deletion from previous backup folders) to be executed.

2) ***This script package does not scale well for***:

a) extra-large files (more than 1GB), which need to be backed up frequently (more than once per day).
b) As a rule of thumb, the size of the data set per differential backup should not exceed on average 2.5% of the total capacity of the backup media. Assumptions are: one differential backup per day (6 days a week), one full backup per week, no **bkdircleanup.sh** usage. Also assumed is that the user keeps the data set generated before last full backup, which consists of previous full backup and all differential backup directories created thereafter. Example:
    - 2TB external HDD fully available for backup (total capacity: 2000GB)
    - full backup on Monday.
    - First full backup after environment setup: 300GB.
    - one differential backup per day from Tuesday till Sunday.
    - Differential backup data set should be maximum 50GB per day.
c) more than 10 users
d) enterprise use (large companies, including point c))
e) databases (e.g. SAP, Oracle), block data
f) any other situation where visible performance decrease is noticed.

3) ***The officially supported OS is OpenSUSE***. The backup & restore package has been tested on and primarily designed for this system. However, as the scripts were written in BASH, they can theoretically be applied on any Unix-like OS. A thorough review and testing is required to check if they properly work on the designated system. The admins should ensure both the command set and used syntax are fully supported and the behavior is as expected. For more details regarding script behavior see logical diagrams contained in this package.

6.3. Additional notes

1) ***In order to get maximum speed and accuracy, it is recommended to logout from any user GUI and enter the system console*** (in OpenSUSE: type CTRL+ALT+F4 to enter, CTRL+ALT+F7 to exit). Get root rights, run the required scripts, then logout and return to the GUI or reboot/shutdown.

2) ***As a best practice, virtual machines and their files should not be included in the home filesystem directory***. Instead, a separate partition or directory should be created for the VM data. We will consider 2 scenarios for the machines that require backup.

Scenario 1: for Linux/Unix-like guests, an instance of each script could be included in each VM. A separate central backup directory would be created for each machine on the backup hard drive. On each VM the scripts will be run individually and completely

independent from the other machines.

Scenario 2: if the guest OS is non-Unix (e.g. Windows), then the scripts are no longer applicable. The specific tools of the guest OS should be used for backup (OS dependent). However, the VM should still be located on a separate partition/in a separate directory to prevent having it backed up by the differential backup script assigned to host OS.

3) ***If system time changes it is highly recommended to do a full backup of the home filesystem***. Not doing this might cause unpredictable behavior of the differential backup script. This note does not apply for restore or cleanup scripts, but only for backup.

4) ***In this documentation, the expressions "home partition" and "home filesystem" were used interchangeably, assuming that the user home directories are located on a dedicated partition***. Although this is strongly recommended, it is not an absolute requirement for running these scripts, which will work even if the user directories are part of a "general purpose" filesystem (for example the root partition). The only difference is that lines 2 and 3 of the Parameters file will not contain mount points but absolute paths of regular directories:

- **Line 2** will contain the absolute path of the parent directory which included the user home folders at previous backup.
- **Line 3** will contain the absolute path of the actual parent directory where all required items will be restored or from where all user items will be backed up. If the mount point of the partition changes same notes will apply as in section 4 – step 9.

Also, the expression “home filesystem directory” was used for naming the parent folder where all user home directories are located. If these are placed on a dedicated home partition, this is equivalent to the partition mount point (e.g. /home).

5) ***In this documentation, the text files are referred to as [file].txt***. This is done for easier identification and understanding of the used meta-data files. The actual files do not have this extension, as it is not required. For example **olddir.txt** has the name **olddir** in the central backup directory (no .txt extension). This should not be changed as it would cause the scripts to stop working or generate errors!

6) ***In this documentation, "items" is a generic word referring to either files or directories.***

APPENDIX A: suggested aliases

Following table presents a list of aliases that could be used for launching the scripts easier. It is assumed that the script executable files are placed in a directory which has the path contained in the environment variable $PATH. If not, the absolute script path should be included in the alias instead of the script name. For example '*su -c /[path]/restoreall.sh*' might be used to launch *restoreall.sh*.

Nr	Alias name	Replaced command	Notes
1	**fbk**	*su -c fbackup.sh*	Full backup only
2	**fbkr**	*su -c 'fbackup.sh; reboot'*	Full backup and reboot
3	**fbks**	*su -c 'fbackup.sh; shutdown -h now'*	Full backup and immediate shutdown
4	**dbk**	*su -c dbackup.sh*	Differential backup only
5	**dbkr**	*su -c 'dbackup.sh; reboot'*	Differential backup and reboot
6	**dbks**	*su -c 'dbackup.sh; shutdown -h now'*	Differential backup and immediate shutdown
7	**rbk**	*su -c rmbkdir.sh*	Remove backup directories created before last full backup
8	**bkdc**	*su -c bkdircleanup.sh*	Remove all m.f.d. items from active backup directories
9	**rsta**	*su -c restoreall.sh*	Restore the entire home filesystem
10	**rstu [username]**	*sudo restoreuser.sh [username]*	Restore user [username]. Command sudo is used instead of su –c, as script contains

			arguments.
11	**rstf [keyword]**	*sudo restorefile.sh [keyword]*	Restore found items by using search keyword [keyword]. Command sudo is used instead of su –c, as script contains arguments.
12	**brparam**	*vi [absolute path of parameters file]*	Edit parameters file using vi

APPENDIX B: script files

File Name	Usage	Located in	Used by	Modified by	Removed after script execution
backup_errors.txt	Contains paths of all items that could not be backed up. Each entry (line) has following format: absolute path of the item as contained in the home filesystem directory.	One instance included in each backup folder where errors occured.	**fbackup.sh, dbackup.sh**	**fbackup.sh, dbackup.sh**	Only if no errors occurred.
backup_operations_log.txt	Same content as **operations_log.txt**. However, while **operations_log.txt** is changed each time a script runs, **backup_operations_log.txt** is not modified anymore after backup finished. It is dedicated to that backup only.	One instance included in each backup folder.	**fbackup.sh, dbackup.sh**	**fbackup.sh, dbackup.sh**	No

backuplist.txt	Contains the names of all active backup directories. First line is always a full backup. Each time a successful differential backup is done, the name of the new backup folder is appended to file. If a new full backup is performed, all lines are erased and replaced with the path of the new backup directory. The erased lines are appended to **old_backuplist.txt**.	Central backup directory	**fbackup.sh, dbackup.sh, restoreall.sh, restoreuser.sh, restorefile.sh, bkdircleanup.sh**	**fbackup.sh, dbackup.sh**	**No**
bdate.txt	At the end of the backup script execution, **temp_bdate.txt** is renamed **bdate.txt**. File **bdate.txt** is used by differential backup scripts to determine which files were kept in the home filesystem AND modified since last backup.	Central backup directory	**fbackup.sh, dbackup.sh**	**fbackup.sh, dbackup.sh**	No

	These files will be backed up.				
deleted.txt	The backup directory cleanup script writes the path of each item which is physically removed from a backup folder to this file. The same path is also removed from the mfd.txt instance located in the same backup folder.	One instance included in each backup folder where m.f.d. files were physically removed.	**bkdircleanup.sh**	**bkdircleanup.sh**	No

erase_errors.txt	Stores the absolute paths of all backup directories that could not be erased by **rmbkdir.sh**. Before actual backup folder removal, the script checks if the file exists. If yes, it will be flushed (emptied). If not, it will be created as empty file. When script execution ends, it is deleted if no errors occurred. If the file had existed prior to running the program, it will only be modified if the script is not stopped by an error (see section 5.2.1.)	Central backup directory	**rmbkdir.sh**	**rmbkdir.sh**	Only if all directories contained in **old_backuplist.txt** have been removed by script.
log.txt	Contains the paths of all items that were backed up AND were not marked for deletion. Paths are relative to home filesystem directory.	One instance included in each backup folder.	**fbackup.sh, dbackup.sh, restoreall.sh, restoreuser.sh, restorefile.sh**	**fbackup.sh, dbackup.sh**	No

mfd.txt	Used by differential backup script to write the paths of items which are marked for deletion. The path of each m.f.d. item is also removed from the **log.txt** instance located in the same backup directory to which the m.f.d. item belongs.	One instance included in each backup folder which contains m.f.d. items.	**dbackup.sh, bkdircleanup.sh**	**dbackup.sh, bkdircleanup.sh**	Only if all m.f.d. items are physically removed from backup directory by **bkdircleanup.sh**.
newdir.txt	Used by differential backup script to list the absolute paths of all items belonging to the home partition. The paths are written to file just before the backup starts. When the backup finishes it is renamed **olddir.txt**.	Central backup directory	**dbackup.sh**	**dbackup.sh**	Renamed at the end of backup

notfound.txt	Stores the paths of all m.f.d. items that could not be found by backup directory cleanup script.	One instance included in each backup folder where m.f.d. items could not be found by **bkdircleanup.sh**	**bkdircleanup.sh**	**bkdircleanup.sh**	No
old_backuplist.txt	Contains the absolute paths of all backup directories created before last full backup. When a new full backup is performed, all paths existing in **backuplist.txt** are appended to **old_backuplist.txt**.	Central backup directory	**fbackup.sh, rmbkdir.sh**	**fbackup.sh, rmbkdir.sh**	Emptied by **rmbkdir.sh**, if script had not been stopped by an error.

olddir.txt	Contains the absolute paths of all items which were belonging to the home filesystem when the previous backup was made. It is updated by each backup script in a different manner: a) **fbackup.sh** writes the item paths to it just before the actual backup begins b) **dbackup.sh** renames **newdir.txt** to **olddir.txt** after all backup operations had finished. File **olddir.txt** is used by differential backup script to determine which item paths were added, removed or kept in the filesystem since previous backup. It is also used by **restorefile.sh** to find the absolute paths of the	Central backup directory	**fbackup.sh, dbackup.sh, restorefile.sh**	**fbackup.sh, dbackup.sh**	No

	items to be restored based on the search keyword.				
operations_log.txt	Detailed log of all operations performed by the last running script. It is overwritten each time a script other than **rmbkdir.sh** is executed.	Central backup directory	**fbackup.sh, dbackup.sh, restoreall.sh, restoreuser.sh, restorefile.sh, bkdircleanup.sh**	**fbackup.sh, dbackup.sh, restoreall.sh, restoreuser.sh, restorefile.sh, bkdircleanup.sh**	No

restore.txt	a) In the user restore scenario, the **log.txt** file is "grepped" for items belonging to the user to be restored. The output is written to **restore.txt**. This operation is performed for each backup directory and only items contained in **restore.txt** are subject to recovery from the current backup folder. File **restore.txt** is thereby modified each time a backup directory is used for item recovery. b) On the other hand, when running **restorefile.sh**, the **olddir.txt** file is "grepped" using the search keyword entered by admin. This happens before the actual file recovery takes place. The found entries are written to	Central backup directory	**restoreuser.sh, restorefile.sh**	**restoreuser.sh, restorefile.sh**	Yes

	restore.txt by keeping only relative path-names to home filesys-tem directory. If item recov-ery is acknowl-edged (user enters 'ok'), the script will start the re-store opera-tion. For each backup direc-tory, only items belong-ing to **re-store.txt** AND **log.txt** will be restored. The content of **re-store.txt** re-mains constant for the whole duration of the **restorefile.sh** script.				

restore_errors.txt	Contains absolute paths of all items that could not be restored by script. Path format is as contained in the backup directory to which the item belongs. If errors occurred during previous restore operations, the file is flushed (made empty) at the start of current restore, otherwise it is created as empty file. At the end of script, it is deleted if no errors occurred.	Central backup directory	**restoreall.sh, restoreuser.sh, restorefile.sh**	**restoreall.sh, restoreuser.sh, restorefile.sh**	Only if no errors occurred
restore_log.txt	Contains the paths of all items that were successfully restored to the home partition. It is used to check if all requested items were recovered. Entry (line) format: relative paths to home filesystem directory.	Central backup directory	**restorefile.sh**	**restorefile.sh**	Yes

restoretemp.txt	Only used in the item restore scenario. For each backup directory the entries belonging to both **log.txt** and **restore.txt** are written to **restoretemp.txt**. Only items belonging to **restoretemp.txt** are recovered.	Central backup directory	**restorefile.sh**	**restorefile.sh**	Yes
temp.txt	Temporary file used for specific operations	Central backup directory	**fbackup.sh, dbackup.sh, restoreall.sh, restoreuser.sh, restorefile.sh, bkdircleanup.sh**	**fbackup.sh, dbackup.sh, restoreall.sh, restoreuser.sh, restorefile.sh, bkdircleanup.sh**	Yes
temp_bdate.txt	This file is created before actual backup begins. When creating the header for **temp_operations_log.txt**, the same output is redirected to **temp_bdate.txt**. At the end of backup it is renamed **bdate.txt**.	Central backup directory	**fbackup.sh, dbackup.sh**	**fbackup.sh, dbackup.sh**	Yes

temp_operations_log.txt	Temporary operations log file created during script runtime. When script finishes execution, the content of the file is overwritten to **operations_log.txt**. If a backup is performed, it will also be overwritten to **backup_operations_log.txt**.	Central backup directory	**fbackup.sh, dbackup.sh, restoreall.sh, restoreuser.sh, restorefile.sh, bkdircleanup.sh**	**fbackup.sh, dbackup.sh, restoreall.sh, restoreuser.sh, restorefile.sh, bkdircleanup.sh**	Yes
temp1.txt	Temporary file used for specific operations.	Central backup directory	**fbackup.sh, dbackup.sh, bkdircleanup.sh, restorefile.sh**	**fbackup.sh, dbackup.sh, bkdircleanup.sh, restorefile.sh**	Yes
tobackup.txt	This file stores the paths which have been added to home filesystem since previous backup. These items will be backed up by **dbackup.sh**. Format of each entry (line) is: relative path to home filesystem directory.	Central backup directory	**dbackup.sh**	**dbackup.sh**	Yes

tocheck.txt	This file stores the paths which have been kept in the home partition since last backup. For each path following applies: a) if item is a directory, it will be backed up by **dbackup.sh** b) if not a directory, it is checked to determine if it has been modified since last backup. c) If yes, it will be backed up. d) If not, no action will be taken. For all backed up items, the instances contained in previous backup directories are marked for deletion. Format of each entry (line) is: relative path to home filesystem directory.	Central backup directory	**dbackup.sh**	**dbackup.sh**	Yes

todelete.txt	This file stores the paths of all items which belong to either category: a) deleted or modified since last backup b) directories with paths belonging to **tocheck.txt**. Format of each entry (line) is: relative path to home filesystem directory. This file is used by dbackup.sh to have all instances of these items marked for deletion from all previous backup folders.	Central backup directory	**dbackup.sh**	**dbackup.sh**	Yes