

Creating a Custom Secure Linux Remove Command with a Trash Bin

presented by

```
animeshsarkar@animeshsarkar:~$ whoami  
animeshsarkar  
animeshsarkar@animeshsarkar:~$
```

High-level design

1. Create a trash directory `~/.trash/`
 2. └— files/ # actual deleted files
 3. └— info/ # metadata (original path, deletion time)
 4. Create a **trash-cleanup** cmd
 5. Demonstration of the trash cleanup script
 6. Create a restore cmd (**srestore**)
 7. Replace rm with our script (**srmv**)
 8. Use a create a cron job for automatic trash clean-up
-

1. Create trash directories

```
animeshsarkar@animeshsarkar:~/animeshsarkar$ mkdir -p ~/.trash/files ~/.trash/info
```

Here, `-p` represents “**parents**”: creates parent directories if not exist; if they do exist it doesn’t give errors.

```
animeshsarkar@animeshsarkar:~/animeshsarkar$ ls -l
total 8
drwxr-xr-x 2 animeshsarkar animeshsarkar 4096 Jan  2 00:34 files
drwxr-xr-x 2 animeshsarkar animeshsarkar 4096 Jan  2 00:34 info
animeshsarkar@animeshsarkar:~/animeshsarkar$
```

2. Ensuring that `~/bin` is in present in **\$PATH**

- Run **echo \$PATH** to print the value of **PATH** variable. Check for “`/home/username/bin`” or “`/Users/username/bin`”. Since mine is set, so I have no reason to worry for now.

```
animeshsarkar@animeshsarkar:~/animeshsarkar$ echo $PATH
/home/animeshsarkar/bin:/home/animeshsarkar/bin:/
```

- If yours not, here’s what you can do. Check your shell by running “**echo \$SHELL**” don’t worry you can run these commands from any location since they are environment variable.

```
animeshsarkar@animeshsarkar:~/animeshsarkar$ echo $SHELL
/bin/bash
animeshsarkar@animeshsarkar:~/animeshsarkar$
```

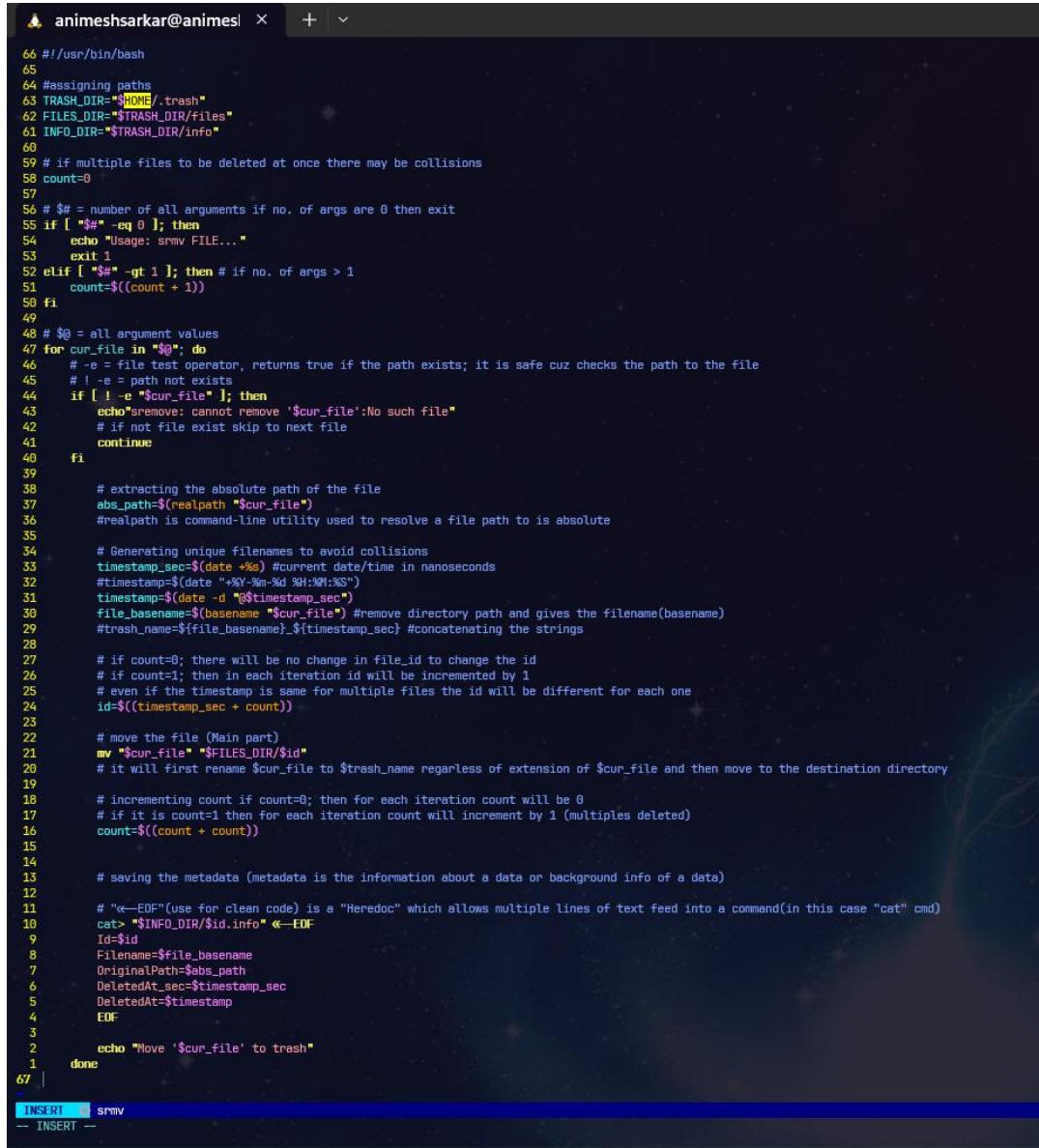
- Mine is **bash** so I need to edit `~/.bashrc` file and add this at the bottom:
`export PATH="$HOME/bin: $PATH"`
- Apply changes: **source ~/.bashrc**
- If the **bin** folder is not created create one **mkdir -p ~/bin**
- Create a test cmd: **echo -e '#!/bin/bash\necho "bin is working fine"' > /bin/testcmd**
- Give executable permission: **chmod +x ~/bin/testcmd**
- Run cmd: **testcmd**. Enjoy.

3. Writing **srmv** cmd script

- Creating the file

```
animeshsarkar@animeshsarkar:~/bin$ nvim srmv
```

- The script



```
#!/usr/bin/bash
#assigning paths
TRASH_DIR="$HOME/.trash"
FILES_DIR="$TRASH_DIR/files"
INFO_DIR="$TRASH_DIR/info"
# if multiple files to be deleted at once there may be collisions
count=0
# $# = number of all arguments if no. of args are 0 then exit
if [ "$#" -eq 0 ]; then
    echo "Usage: srmv FILE..."
    exit 1
elif [ "$#" -gt 1 ]; then # if no. of args > 1
    count=$((count + 1))
fi
# $@ = all argument values
for cur_file in "$@"; do
    # -e = file test operator, returns true if the path exists; it is safe cuz checks the path to the file
    # ! -e = path not exists
    if [ ! -e "$cur_file" ]; then
        echo "remove: cannot remove '$cur_file':No such file"
        # if not file exist skip to next file
        continue
    fi
    # extracting the absolute path of the file
    abs_path=$(realpath "$cur_file")
    #realpath is command-line utility used to resolve a file path to its absolute
    # Generating unique filenames to avoid collisions
    timestamp_sec=$(date +%s) #current date/time in nanoseconds
    timestamp=$(date "+%Y-%m-%d %H:%M:%S")
    timestamp=$(date -d @"$timestamp_sec")
    file_basename=${abs_path##*/}
    file_basename=${file_basename%.*} #remove directory path and gives the filename(basename)
    trash_name=${file_basename}_${timestamp} #concatenating the strings
    # if count=0; there will be no change in file_id to change the id
    # if count=1; then in each iteration id will be incremented by 1
    # even if the timestamp is same for multiple files the id will be different for each one
    id=$((timestamp_sec + count))
    # move the file (Main part)
    mv "$cur_file" "$FILES_DIR/$id"
    # it will first rename $cur_file to $trash_name regardless of extension of $cur_file and then move to the destination directory
    # incrementing count if count=0; then for each iteration count will be 0
    # if it is count=1 then for each iteration count will increment by 1 (multiples deleted)
    count=$((count + count))
    # saving the metadata (metadata is the information about a data or background info of a data)
    # "<--EOF" (use for clean code) is a "Heredoc" which allows multiple lines of text feed into a command(in this case "cat" cmd)
    cat> "$INFO_DIR/$id.info" <--EOF
    id=$id
    Filename=$file_basename
    OriginalPath=$abs_path
    DeletedAt_sec=$timestamp_sec
    DeletedAt=$timestamp
    EOF
done
echo "Move '$cur_file' to trash"
done

```

- Giving executable permission: **chmod a+x srmv** (giving everyone executable permission)

```
animeshsarkar@animeshsarkar:~/bin$ ls -l srmv
-rw-r--r-- 1 animeshsarkar animeshsarkar 1539 Jan  2 00:57 srmv
animeshsarkar@animeshsarkar:~/bin$ chmod a+x srmv
animeshsarkar@animeshsarkar:~/bin$ ls -l srmv
-rwxr-xr-- 1 animeshsarkar animeshsarkar 1539 Jan  2 00:57 srmv
animeshsarkar@animeshsarkar:~/bin$
```

4. Clean-up script

```

4 FILES_DIR="$TRASH_DIR/files"
1 INFO_DIR="$TRASH_DIR/info"
2
3 # current time
4 cur_timestamp=$(date +%s)
5
6 # maximum age of a target (120 days) in seconds i.e. 10368000 sec
7 # max_age=$((120 * 24 * 60 * 60))
8 max_age=$(( 2 * 60 )) # for testing current it's 2 mins
9
10 # traversing over the info folder to determine if any target is aged for clean-up
11 # * → wild card; *.info → anytarget with ".info"
12 for target in "$FILES_DIR"/*; do
13     # checking if any info target exist or not using target test operator(-e)
14     # if path not exists [ -e "$info" ] will return false
15     [ -e "$target" ] || continue # false ||(OR) continue → skip to next target
16
17     # target = /home/animeshsarkar/.trash/files/1768058958
18     target_id=$(basename "$target") # 1768058958 (basename)
19     info="$INFO_DIR/$target_id.info" # /home/animeshsarkar/.trash/info/1768058958.info
20
21     # else extract the deleted time in sec
22     deletedAt_sec=$(awk -F= '$1=="DeletedAt_sec" && $2 ~ /^[0-9]+$/ { print $2 }' "$info")
23
24     # using "awk" cmd for extracting;
25     # -F→ field separator; here "equals to(=)" serves as field separator
26     # $1→ field 1; $2→field 2; ... $n→field n;
27     # $1="DeletedAt_sec"; $2= is the timestamp(170001234)
28     # 170001234→^(starting with); [0-9]+(one or more(plus sign "+") digits); $(ends with)
29     # print $2 → print field 2
30     # $info → the target
31
32     if (( cur_timestamp-deletedAt_sec > max_age )); then
33         # flag "-f" for force remove (no prompts)
34         # flag "-r" for recursive for deleting folders
35
36         # flag "-d" checks if the target is a directory
37         if [ -d "$target" ]; then
38             rm -rf "$target"
39         else # the target is a file
40             rm -f "$target"
41         fi
42
43         rm -f "$info"
44     fi
45
46 done
47
48
-- INSERT -- .. /bin/trash-cleanup
-- INSERT --

```

5. Create an interactive restore cmd (*srestore*)

- Show all the trash file info with **index**

```

1#!/usr/bin/bash
1
2 FILES_DIR="$HOME/.trash/files"
3 INFO_DIR="$HOME/.trash/info"
4
5 # Exit if no files (safe)
6 shopt -s nullglob
7
8 # Header
9 printf "%-5s %-20s %-22s %\n" "Index" "ID" "DeletedAt" "OriginalPath"
10 printf "%-5s %-20s %-22s %\n" "----" "----" "-----" "-----"
11
12 index=1
13
14 # Loop over .info files sorted numerically, newest first
15 printf '%s\n' "$INFO_DIR"/*.info |
16 sort -nr |
17 while IFS= read -r info; do # here is info is the variable where the input value is stored
18     item_info=$(basename "$info")
19
20     id=$(awk -F= '$1=="Id" && $2 ~ /^[0-9]+$/ { print $2 }' "$info")
21
22     delete_loc=$(awk -F= '$1=="OriginalPath" { print $2 }' "$info")
23
24     delete_time=$(awk -F= '$1=="DeletedAt_sec" && $2 ~ /^[0-9]+$/ { print $2 }' "$info")
25     # convert into human readable
26     deleted_human=$(date -d "@$delete_time" "+%Y-%m-%d %H:%M:%S")
27
28     printf "%-5d %-20s %-22s %\n" "$index" "$id" "$deleted_human" "$delete_loc"
29
30     index=$((index + 1))
31 done
32

```

- Taking user prompt to restore the file via index number

```

43 # user prompt
42 read -p "Enter the index of the file to restore: " choice
41
40 # choice should be a valid number
39 if ! [[ "$choice" =~ ^[0-9]+$ ]]; then
38   echo "Invalid input"
37   exit 1
36 fi
35
34 index=1
33
32 printf '%s\n' "$INFO_DIR"/*.info |
31 sort -nr |
30 while IFS= read -r info; do
29   # if index is not e
28   if [ "$index" -ne "$choice" ]; then
27     index=$((index + 1))
26     continue
25   fi
24
23
22 #item_info=$(basename "$info")
21
20 id=$(awk -F= '$1=="Id" && $2 ~ /^[0-9]+$/ { print $2 }' "$info")
19
18 original_path=$(awk -F= '$1=="OriginalPath" { print $2 }' "$info")
17
16 if [ ! -e "$FILES_DIR/$id" ] ; then
15   echo "Sorry, file not found."
14   exit 1
13 fi
12
11 # incase if the directory or the path is removed create one
10 mkdir -p "$(dirname "$original_path")"
9
8 # renaming and moving the file to it's original location
7 mv "$FILES_DIR/$id" "$original_path"
6 rm -f "$info"
5
4 echo "Restored: $original_path"
3 break
2
1 done
77
INSERT ↵ ./bin/srestore
-- INSERT --

```

- Here is tip the moment your code works. Save it in some other folder as the quick backup. I overwrote the **srestore** cmd but luckily I had the backup file.

```

animeshsarkar@animeshsarkar:~/bin$ cp ../Backup/Scripts/srestore srestore
animeshsarkar@animeshsarkar:~/bin$ nvim srestore

```

6. Final Demonstration

- Creating test files and directories

```

animeshsarkar@animeshsarkar:~/Test$ mkdir -p DeleteMe
animeshsarkar@animeshsarkar:~/Test$ nvim DeleteMe/test1.txt
animeshsarkar@animeshsarkar:~/Test$ nvim test2.c
animeshsarkar@animeshsarkar:~/Test$ ls
DeleteMe qweoiqjot.txt test2.c
animeshsarkar@animeshsarkar:~/Test$ ls DeleteMe/
test1.txt
animeshsarkar@animeshsarkar:~/Test$ 

```

- Using our custom **srmv** cmd

```

animeshsarkar@animeshsarkar:~/Test$ srmv DeleteMe test2.c
Move 'DeleteMe' to trash
Move 'test2.c' to trash
animeshsarkar@animeshsarkar:~/Test$ 

```

- Verifying if the items moved to **.trash**

```
animeshsarkar@animeshsarkar:~/Test$ ls ../.trash/files/
1768152296 1768153430 1768162314 1768164764 1768164766
animeshsarkar@animeshsarkar:~/Test$ ls ../.trash/info
1768152296.info 1768153430.info 1768162314.info 1768164764.info 1768164766.info
animeshsarkar@animeshsarkar:~/Test$
```

- Restore file and directory. **Data is intact.**

```
animeshsarkar@animeshsarkar:~/Test$ srestore
Index ID          DeletedAt        OriginalPath
-----
1    1768165254   2026-01-12 02:30:52  /home/animeshsarkar/Test/test2.c
2    1768165253   2026-01-12 02:30:52  /home/animeshsarkar/Test/DeleteMe
3    1768165112   2026-01-12 02:28:32  /home/animeshsarkar/bin/trash-list
4    1768165112   2026-01-11 23:13:50  /home/animeshsarkar/Test/nfsioadsgfsg.txt
5    1768165112   2026-01-11 22:54:56  /home/animeshsarkar/Test/nfsioasg.txt

Enter the index of the file to restore: 1
Restored: /home/animeshsarkar/Test/test2.c
animeshsarkar@animeshsarkar:~/Test$ cat test2.c
int main() {
    return 0;
}
animeshsarkar@animeshsarkar:~/Test$ srestore
Index ID          DeletedAt        OriginalPath
-----
1    1768165253   2026-01-12 02:30:52  /home/animeshsarkar/Test/DeleteMe
2    1768165112   2026-01-12 02:28:32  /home/animeshsarkar/bin/trash-list
3    1768165112   2026-01-11 23:13:50  /home/animeshsarkar/Test/nfsioadsgfsg.txt
4    1768165112   2026-01-11 22:54:56  /home/animeshsarkar/Test/nfsioasg.txt

Enter the index of the file to restore: 1
Restored: /home/animeshsarkar/Test/DeleteMe
animeshsarkar@animeshsarkar:~/Test$ ls
DeleteMe qweoiqjot.txt test2.c
animeshsarkar@animeshsarkar:~/Test$ cd DeleteMe/
animeshsarkar@animeshsarkar:~/Test/DeleteMe$ ls
test1.txt
animeshsarkar@animeshsarkar:~/Test/DeleteMe$
```

- Creating and removing a new file whose max age is less than 2 min

```
animeshsarkar@animeshsarkar:~/Test$ echo "$(date '+%Y-%m-%d %H:%M:%S')" > testmeagain.txt
animeshsarkar@animeshsarkar:~/Test$
```

```
animeshsarkar@animeshsarkar:~/Test$ srestore
Index ID          DeletedAt        OriginalPath
-----
1    1768166694   2026-01-12 02:54:54  /home/animeshsarkar/Test/DeleteMe
2    1768166683   2026-01-12 02:54:54  /home/animeshsarkar/Test/test3.txt
3    1768166681   2026-01-12 02:54:54  /home/animeshsarkar/Test/test2.txt
4    1768166680   2026-01-12 02:54:39  /home/animeshsarkar/Test/test1.txt

Enter the index of the file to restore: 
```

- **trash-cleanup** the items which are older than assigned age i.e. 2 min for this case

```
animeshsarkar@animeshsarkar:~/Test$ rmv testmeagain.txt
Move 'testmeagain.txt' to trash
animeshsarkar@animeshsarkar:~/Test$ srestore
Index ID          DeletedAt        OriginalPath
-----
1    1768166932   2026-01-12 02:58:52  /home/animeshsarkar/Test/testmeagain.txt
2    1768166694   2026-01-12 02:54:54  /home/animeshsarkar/Test/DeleteMe
3    1768166683   2026-01-12 02:54:39  /home/animeshsarkar/Test/test3.txt
4    1768166681   2026-01-12 02:54:39  /home/animeshsarkar/Test/test2.txt
5    1768166680   2026-01-12 02:54:39  /home/animeshsarkar/Test/test1.txt

Enter the index of the file to restore: ^C
animeshsarkar@animeshsarkar:~/Test$ trash-cleanup
animeshsarkar@animeshsarkar:~/Test$ srestore
Index ID          DeletedAt        OriginalPath
-----
1    1768166932   2026-01-12 02:58:52  /home/animeshsarkar/Test/testmeagain.txt

Enter the index of the file to restore: 
```

7. Alias rm

```
121 # Alias of rm cmd
 1 alias rm='rmv'
-
-
-
NORMAL .bashrc[+]
```

Still can use **rm** via: `\rm filename`

8. Schedule clean-up script (**use at your own risk**)

Will do later need some extra config to set up. If you want to setup just google it. And if you have a minute, just read the below explanation, it's awesome.

- Using **crontab -e** (Cron jobs are executed by the cron daemon, which runs in the background independent of the shell and continues to run even when the user is logged out)

```
└ minute (0–59)
  └ hour (0–23)
    └ day of month (1–31)
      └ month (1–12)
        └ day of week (0–7)
          └
          └
```

*** * * * * command (absolute path)**

How I remember this thing:

- First of all, **Wildcard character (*)**. It means **everything**.

- Now, **1 * * * *** means:

```
1 = at 1 min
* = every hour of the day
* = every day of the month
* = every month of the year
* = every day of the week
```

For example: the script will run, at 00:01, then 01:01, then 02:01 ... 23:01.

And then repeat the next day.

- Common schedules

0 * * * * = Every hour (at the top of the hour). Like 17:00, 18:00, 00:00 etc.

0 0 * * * = Every night at midnight. Like Monday at 00:00, Tuesday at 00:00 etc.

0 = 0th min

0 = 0th hour i.e. at midnight

*** * *** = every day; every month; every day of the week

***/15 * * * *** = Every 15 minutes (the / means "increment").

23 9-17 * * 1-5 = Every 23rd min on the hour, but only between 9 AM and 5 PM, and only on weekdays (Monday-Friday).

Like on Monday at 09:23, 10:23 ...17:23, on Tuesday at 09:23, 10:23 ...17:23, ... Friday at 09:23, 10:23 ...17:23. But will not be executed on weekends. It's their holiday.