

Creating a Custom Secure Linux Remove Command with a Trash Bin

```
1 #!/usr/bin/bash
2 #assigning paths
3 TRASH_DIR="$HOME/.trash"
4 FILES_DIR="$TRASH_DIR/files"
5 INFO_DIR="$TRASH_DIR/info"
6
7 # if multiple files to be deleted at once there may be collisions
8 count=0
9
10 # $# = number of all arguments if no. of args are 0 then exit
11 if [ "$#" -eq 0 ]; then
12     echo "Usage: srwv FILE..."
13     exit 1
14 elif [ "$#" -gt 1 ]; then # if no. of args > 1
15     count=$((count + 1))
16 fi
17
18 # $@ = all argument values
19 for cur_file in "$@"; do
20     # -e = file test operator, returns true if the path exists; !
21     # ! -e = path not exists
22     if [ ! -e "$cur_file" ]; then
23         echo "remove: cannot remove '$cur_file': No such file"
24         # if not file exist skip to next file
25         continue
26     fi
27
28     # extracting the absolute path of the file
29     abs_path=$(realpath "$cur_file")
30     #realpath is command-line utility used to resolve a file
31     # path to its absolute
32
33     # Generating unique filenames to avoid collisions
34     timestamp_sec=$(date +%s) #current date/time in nanosecond
35     ds
36     timestamp=$(date -d "$timestamp_sec" +%Y-%m-%d %H:%M:%S)
37     file_basename=$(basename "$cur_file") #remove directory p
38     # path and gives the filename(basename)
39     # #trash_name=$(file_basename).$(timestamp_sec) #concatenat
40     # ing the strings
41
42     # if count=0; there will be no change in file_id to chang
43     # e the id
44     # if count=1; then in each iteration id will be incremen
45     # ed by 1
46     # even if the timestamp is same for multiple files the
47     # id will be different for each one
48     id=$((timestamp_sec + count))
49
50     # Index ID: 1/67454
51     # bin/srwv
52
53     #!/usr/bin/bash
54     2 FILES_DIR="$HOME/.trash/files"
55     3 INFO_DIR="$HOME/.trash/info"
56     4 # Exit if no files (safe)
57     5 shopt -s nullglob
58     6
59     7 # Header
60     8 printf "%-5s %-20s %-22s %s\n" "Index" "ID" "DeletedAt" "Original
61     Path"
62     9
63     10 printf "%-5s %-20s %-22s %s\n" "-----" "-----" "-----" "-----"
64     11
65     12 # Loop over .info files sorted numerically, newest first
66     13 index=1
67     14 while IFS= read -r info; do # here is info is the variable where
68         # the input value is stored
69         15 id=$(awk -F "$1=" '{print $2}' "$info")
70         16 item_info=$(basename "$info")
71         17 delete_loc=$(awk -F "$1=" '{print $2}' "$info")
72         18
73         19 delete_time=$(awk -F "$1=" '{print $2}' "$info" | sed 's/^[0-9]*$//')
74         20 # convert into human readable
75         21 deleted_human=$(date -d "$delete_time" +%Y-%m-%d %H:%M:%S)
76         22 printf "%-5d %-20s %-22s %s\n" "$index" "$id" "$deleted_human"
77         23 "$delete_loc"
78         24 index=$((index + 1))
79     done
80
81     25 # user prompt
82     26 read -p "Enter the index of the file to restore: " choice
83     27
84     28 # choice should be a valid number
85     29 if [ ! -z "$choice" ]; then
86         30 echo "Invalid input"
87         31 exit 1
88     fi
89
90     32 bin/srestore
91
92     #!/usr/bin/bash
93     1 TRASH_DIR="$HOME/.trash"
94     2 FILES_DIR="$TRASH_DIR/files"
95     3 INFO_DIR="$TRASH_DIR/info"
96     4
97     5 # current time
98     6 cur_timestamp=$(date +%s)
99     7
100    8 # maximum age of s target (120 days) in seconds i.e. 1036800 sec
101    9 max_age=$((120 * 24 * 60 * 60))
102    10
103    11 # traversing over the info folder to determine if any target is o
104    12 # ed for clean-up
105    13 # * -> wild card; *.info -> anytarget with ".info"
106    14 for target in "$FILES_DIR"/*.info; do
107        15 # checking if any info target exist or not using target test
108        16 operator(-e)
109        17 # if path not exists [ -e "$info" ] will return false
110        18 [ -e "$target" ] || continue # false || (OR) continue -> skip
111        19 to next target
112
113        20 # target = /home/animeshsarkar/.trash/files/1768058958
114        21 target_id=$(basename "$target") # 1768058958 (basename)
115        22 info=$(INFO_DIR/$target_id.info) # /home/animeshsarkar/.trash
116        23 /info/1768058958.info
117
118        24 # also extract the deleted time in sec
119        25 deleted_at_sec=$(awk -F "$1=" '{print $2}' "$info")
120        26 # $/ { print $2 }' "$info"
121
122        27 # using "awk" cmd for extracting;
123        28 # -F -> field separator; here "equals to(=)" serves as field s
124        25
126        29 # $1 -> field 1; $2 -> field 2; ... $n -> field n;
127        30 # $1="DeletedAt_sec"; $2 is the timestamp(176801234)
128        31 # 176801234 -> (starting with); [0-9]+(one or more plus sign "
129        32 +") digits); $(ends with)
130        31 # print $2 -> print field 2
131        32 # $info -> the target
132
133        33 if (( cur_timestamp-deleted_at_sec > max_age )); then
134            34 # flag "-f" for force remove (no prompts)
135            35 # flag "-r" for recursive for deleting folders
136            36 # flag "-d" checks if the target is a directory
137            37 if [ -d "$target" ]; then
138                38 rm -rf "$target"
139            else # the target is a file
140                39
141            fi
142        fi
143    done
```

```
1 #assigning paths
2 TRASH_DIR="$HOME/.trash"
3 FILES_DIR="$TRASH_DIR/files"
4 INFO_DIR="$TRASH_DIR/info"
5
6 # if multiple files to be deleted
7 count=0
8
9 # $# = number of all arguments if
10 if [ "$#" -eq 0 ]; then
11     echo "Usage: srwv FILE..."
12     exit 1
13 elif [ "$#" -gt 1 ]; then # if no.
14     count=$((count + 1))
15 fi
16
17 # $@ = all argument values
18 for cur_file in "$@"; do
19     # -e = file test operator, ret
20     # ! -e = path not exists
21     if [ ! -e "$cur_file" ]; then
22         echo "remove: cannot remov
23         # if not file exist skip
24         continue
25     fi
26
27     # extracting the absolute
28     abs_path=$(realpath "$cur
29     #realpath is command-line
30
31     # Generating unique filena
32     timestamp_sec=$(date +%s)
33     timestamp=$(date +%Y-%m-%d %H:%M:%S)
34     file_basename=$(basename "$cur_file")
35     #trash_name=$(file_basename).$(timestamp_sec)
36
37     # if count=0; there will be no change in file_id to change the id
38     # if count=1; then in each iteration id will be incremented by 1
39     # even if the timestamp is same for multiple files the id will be different for each one
40     id=$((timestamp_sec + count))
41
42     # Index ID: 1/67454
43     # bin/srwv
44
45     #!/usr/bin/bash
46     2 1768166683 2026-01-12 02:54:39 /home/animeshsarkar/Test/test3.txt
47     3 1768166681 2026-01-12 02:54:39 /home/animeshsarkar/Test/test2.txt
48     4 1768166680 2026-01-12 02:54:39 /home/animeshsarkar/Test/test1.txt
49
50     Enter the index of the file to restore: 4C
51     animeshsarkar@animeshsarkar:~/Test$ echo "$(date +%Y-%m-%d %H:%M:%S)" > testneagain.txt
52     animeshsarkar@animeshsarkar:~/Test$ mvim ../bin/trash-cleanup
53     animeshsarkar@animeshsarkar:~/Test$ srwv testneagain.txt
54     Move 'testneagain.txt' to trash
55     animeshsarkar@animeshsarkar:~/Test$ srestore
56
57     Index ID DeletedAt OriginalPath
58     -----
59     1 1768166932 2026-01-12 02:58:52 /home/animeshsarkar/Test/testneagain.txt
60     2 1768166694 2026-01-12 02:54:54 /home/animeshsarkar/Test/Deleteme
61     3 1768166653 2026-01-12 02:54:39 /home/animeshsarkar/Test/test3.txt
62     4 1768166681 2026-01-12 02:54:39 /home/animeshsarkar/Test/test2.txt
63     5 1768166680 2026-01-12 02:54:39 /home/animeshsarkar/Test/test1.txt
64
65     Enter the index of the file to restore: 4C
66     animeshsarkar@animeshsarkar:~/Test$ trash-cleanup
67     animeshsarkar@animeshsarkar:~/Test$ srestore
68
69     Index ID DeletedAt OriginalPath
70     -----
71     1 1768166932 2026-01-12 02:58:52 /home/animeshsarkar/Test/testneagain.txt
72
73     Enter the index of the file to restore: 4C
74     animeshsarkar@animeshsarkar:~/Test$ mvim ../bin/srestore
75     animeshsarkar@animeshsarkar:~/Test$ mvim ../bin/trash-cleanup
76     animeshsarkar@animeshsarkar:~/Test$ mvim ../bin/trash-cleanup
77     animeshsarkar@animeshsarkar:~/Test$ cd
78     animeshsarkar@animeshsarkar:~/Test$ ls
79     Backup C-genes Test bin srwv_1767468454 tetris.c tetrisl tetris2.c trash-cleanup.log trash-cleanup_1767468454 trash-n
80     ew.log trash.log
81     animeshsarkar@animeshsarkar:~/Test$ ls -la
82     animeshsarkar@animeshsarkar:~/Test$
83
84     #trash_name=$(file_basename).$(timestamp_sec)
85
86     # if count=0; there will be no change in file_id to change the id
87     # if count=1; then in each iteration id will be incremented by 1
88     # even if the timestamp is same for multiple files the id will be different for each one
89     id=$((timestamp_sec + count))
90
91     # Index ID: 1/67454
92     # bin/srwv
93
94     NORMAL bin/srwv
```

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:~$ 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:~/.trash$ 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:~/.trash$
```

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

- 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\nnecho "bin is working fine"' > /bin/testcmd`**
- Give executable permission: **`chmod +x ~/bin/testcmd`**
- Run cmd: **`testcmd`**. Enjoy.

3. Writing *rmv* cmd script

- Creating the file

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

- The script

```
animeshsarkar@animes! x + v
66 #!/usr/bin/bash
67
68 #assigning paths
69 TRASH_DIR=~/.trash
70 FILES_DIR=~/.trash/files
71 INFO_DIR=~/.trash/info
72
73 # if multiple files to be deleted at once there may be collisions
74 count=0
75
76 # $# = number of all arguments if no. of args are 0 then exit
77 if [ "$#" -eq 0 ]; then
78     echo "Usage: srmv FILE..."
79     exit 1
80 fi
81 elif [ "$#" -gt 1 ]; then # if no. of args > 1
82     count=$((count + 1))
83 fi
84
85 # $@ = all argument values
86 for cur_file in "$@"; do
87     # -e = file test operator, returns true if the path exists; it is safe cuz checks the path to the file
88     # ! -e = path not exists
89     if [ ! -e "$cur_file" ]; then
90         echo "remove: cannot remove '$cur_file': No such file"
91         # if not file exist skip to next file
92         continue
93     fi
94
95     # extracting the absolute path of the file
96     abs_path=$(realpath "$cur_file")
97     #realpath is command-line utility used to resolve a file path to its absolute
98
99     # Generating unique filenames to avoid collisions
100     timestamp_sec=$(date +%s) #current date/time in nanoseconds
101     #timestamp=$(date +%Y-%m-%d %H:%M:%S)
102     timestamp=$(date -d "@$timestamp_sec")
103     file_basename=$(basename "$cur_file") #remove directory path and gives the filename(basename)
104     #trash_name=${file_basename}_${timestamp_sec} #concatenating the strings
105
106     # if count=0; there will be no change in file_id to change the id
107     # if count=1; then in each iteration id will be incremented by 1
108     # even if the timestamp is same for multiple files the id will be different for each one
109     id=$((timestamp_sec + count))
110
111     # move the file (Main part)
112     mv "$cur_file" "$FILES_DIR/$id"
113     # it will first rename $cur_file to $trash_name regardless of extension of $cur_file and then move to the destination directory
114
115     # incrementing count if count=0; then for each iteration count will be 0
116     # if it is count=1 then for each iteration count will increment by 1 (multiples deleted)
117     count=$((count + count))
118
119     # saving the metadata (metadata is the information about a data or background info of a data)
120
121     # "«-EOF"(use for clean code) is a "Heredoc" which allows multiple lines of text feed into a command(in this case "cat" cmd)
122     cat> "$INFO_DIR/$id.info" «-EOF
123     Id=$id
124     Filename=$file_basename
125     OriginalPath=$abs_path
126     DeletedAt_sec=$timestamp_sec
127     DeletedAt=$timestamp
128     EOF
129
130     echo "Move '$cur_file' to trash"
131 done
132
```

- 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-x 1 animeshsarkar animeshsarkar 1539 Jan  2 00:57 srmv
animeshsarkar@animeshsarkar:~/bin$
```


4. Clean-up script

```

animeshsarkar@animesh x animeshsarkar@animesh x + v
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     fi
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

```

animeshsarkar@animesh x animeshsarkar@animesh x + v
1 !/usr/bin/bash
2
3 FILES_DIR="$HOME/.trash/files"
4 INFO_DIR="$HOME/.trash/info"
5 # Exit if no files (safe)
6 shopt -s nullglob
7
8 # Header
9 printf "%-5s %-20s %-22s %s\n" "Index" "ID" "DeletedAt" "OriginalPath"
10 printf "%-5s %-20s %-22s %s\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 %s\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

```

animeshsarkar@animeshl x animeshsarkar@animeshl x + ~
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' "$INFO_DIR"/*.info |
31 sort -nr |
30 while IFS= read -r info; do
29     # if index is not 0
28     if [ "$index" -ne "$choice" ]; then
27         index=$((index + 1))
26         continue
25     fi
24
23     #item_info=$(basename "$info")
22
21     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          2026-01-11 23:13:50 /home/animeshsarkar/Test/nfsioadsgfsg.txt
5          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          2026-01-11 23:13:50 /home/animeshsarkar/Test/nfsioadsgfsg.txt
4          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 02:54:39 /home/animeshsarkar/Test/test3.txt
3 1768166681      2026-01-12 02:54:39 /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: 1
```

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

```
animeshsarkar@animeshsarkar:~/Test$ srmv 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: 1
```

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.

00***** = 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.