Commento a `trash.sh`

I comandi saranno interpretati dalla *shell* **/bin/sh**; vogliamo vedere la traccia dei comandi eseguiti (**-x**).

```
#!/bin/sh -x
#
# trash.sh
#
# moves file(s) to trash directory
# DEMO/TUTORIAL ONLY -- DO NOT USE IN ANY OTHER ENVIRONMENT
#
# pc20050217

```

Se il numero di parametri (**$#**) passati allo script è zero, allora:
scopri il nome con cui è stato chiamato lo script e assegnalo alla variabile **self**
visualizza un messggio che spieghi come va usato lo script
esci, segnalando la condizione di errore

```
if [ $# -eq 0 ] ; then
 self=$(basename $0)
 echo "Usage: $self filename [filename...]"
 exit 1
fi

```

Definisci il nome del file di configurazione e assegnalo alla variabile **$trashrc**: ogni utente può usare un configurazione differente, quindi il file di configurazione sta nella sua home directory. Se il file esiste (**-r**) allora leggilo ed esegui i comandi in esso contenuti...

```
trashrc="${HOME}/.trashrc"
if [ -r $trashrc ] ; then
 . $trashrc
```

...altrimenti scrivi un nuovo file di configurazione con dei valori di *default*. Qui usiamo l'operatore di redirezione dell'output **>** per scrivere nel file e il costrutto *here-document* per creare il contenuto: tutto ciò che è contenuto tra **<<-EOF** e la riga con solamente **EOF** viene passato come input a **cat**; lo spazio all'inizio di ogni riga viene eliminato. Poi, come sopra, esegui i comandi di configurazione.

```
else
 cat > $trashrc <<-EOF
        TRASHDIR="${HOME}/.trash.d"
        TRASHINFO="${HOME}/.trash.d/.trashinfo"
        EOF
 . $trashrc
fi

```

(da qui vediamo anche che il file di configurazione contiene semplicemente la definizione di due variabili)

Se non esiste la directory «cestino», creala. Se qualcosa va storto, **mkdir** restituisce in **$?** un valore diverso da zero: significa che non è stato possibile creare la directory e ci fermiamo qui, propagando la condizione di errore.

```
if [ ! -d ${TRASHDIR} ] ; then
 mkdir ${TRASHDIR}
 if [ $? -ne 0 ] ; then
  echo "mkdir ${TRASHDIR} failed!"
  exit 2
 fi
fi

```

Inizio del ciclo che sposta nel cestino, uno alla volta, i file dati come parametri allo script. La condizione di uscita dal ciclo è che il primo parametro sia una stringa nulla, cioè che non ci siano (più) parametri disponibili (**-z**)

```
until [ -z $1 ] ; do
```

Assegna alla variabile **file** il nome del primo file su cui operare; quindi controlla che sia veramente un file (**-f**) e non ad es. una directory, o un nome di un file inesistente: se non è un file, esci subito, segnalando la condizione di errore.

```
36    file=$1
37    if [ ! -f ${file} ] ; then
38     echo "${file} is not a regular file!"
39     exit 3
40    fi
```

Estrai il nome del file, eliminando eventuali nomi di directory che lo precedono, e assegnalo alla variabile **`filename`**. Quindi crea un nuovo file nella directory «cestino», con un nome basato sul nome del file che vogliamo cestinare. Infine scrivi in un registro la directory in cui stiamo lavorando, il nome del file cestinato e il nome con cui è stato salvato nel cestino.

```
41    filename=$(basename $1)
42    junkfile=$(mktemp ${TRASHDIR}/${filename}_XXXXXX)
43    junkname=$(basename $junkfile)
44    echo -ne "$(pwd)\t${file}\t${junkname}\n" >> ${TRASHINFO}
45
```

Infine sposta il file nel cestino, con il nuovo nome creato sopra

```
46    mv ${file} ${junkfile}
```

Per proseguire con gli (eventuali) altri altri file passati come parametri allo script, rinomina i parametri, in modo che il primo (che abbiamo già usato) sparisca, il secondo diventi il primo, ecc. Quindi chiudi il ciclo.

```
47    shift
48   done
49
50   #EOF
```

Esempio di esecuzione

`./trash.sh testfile`

Ogni comando eseguito direttamente dalla shell è preceduto da **`+`** , i comandi eseguiti indirettamente sono preceduti da **`++`**

Riga 10: **`$#`** è 1:

```
+ '[' 1 -eq 0 ']'
```

Riga 16: **`$HOME`** è `/home/piero`

```
+ trashrc=/home/piero/.trashrc
+ '[' -r /home/piero/.trashrc ']'
```

Riga 18: esegui il contenuto del file di configurazione

```
+ . /home/piero/.trashrc
++ TRASHDIR=/home/piero/.trash.d
++ TRASHINFO=/home/piero/.trash.d/.trashinfo
```

Riga 27: la directory esiste già

```
+ '[' '!' -d /home/piero/.trash.d ']'
```

Righe 35–43: assegna alle variabili i nomi dei file su cui operare, verifica che il file esista, crea il nuovo file nel cestino

```
+ '[' -z testfile ']'
+ file=testfile
+ '[' '!' -f testfile ']'
++ basename testfile
+ filename=testfile
++ mktemp /home/piero/.trash.d/testfile_XXXXXX
+ junkfile=/home/piero/.trash.d/testfile_yU6323
++ basename /home/piero/.trash.d/testfile_yU6323
+ junkname=testfile_yU6323
++ pwd
```

Righe 44–46: scrivi una riga nel registro e sposta il file

```
+ echo -ne '/home/piero/data/Galvani2005\ttestfile\ttestfile_yU6323\n'
+ mv testfile /home/piero/.trash.d/testfile_yU6323
```

Righe 47 e 35: sposta i parametri; non ci sono altri parametri, finiamo qui

```
+ shift
+ '[' -z ']'
```