Linux Shell Scripting - Delete User Accounts

In this tutorial I will be creating a shell script that allows for a local Linux account to be disabled, deleted, and optionally archived.

Prerequisites

- access to a running Linux distribution
- local user accounts that can be disabled/deleted

To complete this tutorial, you will need access to a running Linux distribution. If you do not already have access to a Linux system, I have a number of VirtualBox tutorials where I demonstrate the installation of CentOS 7 and Ubuntu 22 as virtual machines, accessible **here**.

I also have a few tutorials where I demonstrate the creation of AWS compute instances, RHEL 8 and Ubuntu 20, accessible **here**. Keep in mind that you will need to create an AWS account to be able to create a compute instance. If you do not have an AWS account, my tutorial **Create AWS Free Tier Account** is accessible **here.**

For this tutorial, I will be using my **CentOS 7 VM** for testing, but the script will work on any running Linux system.

At the beginning of the tutorial, I will also be providing a script that creates test accounts.

This script will, by default, disable a user account on the local system. The person executing this script must have superuser privileges. When executing the script, at least one username argument must be supplied (user account(s) with UID >= 1000). Three options can also be used when executing the script:

-d Deletes accounts instead of disabling them.
-r Removes the home directory associated with the account(s).
-a Creates an archive of the home directory associated with the account(s).

Finally, it will display the account username and any actions performed on the account(s).

Steps to complete tutorial:

Create Test Accounts

In my home directory, I created a new **scripts** directory and, in it, I placed a newly created script file named **add_test_accounts.sh** with the following contents.

```
# Add some accounts to test with
if [[ "${UID}" -ne 0 ]]
then
  echo "You must use superuser privileges to run this script." >&2
  exit 1
fi

for U in fredj sallys archieb earlp bertl
do
  useradd ${U}
  echo 'pass123' | passwd --stdin ${U}
done
```

I will now execute the script to create some test accounts which will be used during the tutorial.

```
$ chmod +x add_test_accounts.sh
$ sudo ./add_test_accounts.sh
$ ls -l /home
```

```
[liam@centos7-vm scripts]$
[liam@centos7-vm scripts]$ chmod +x add_test_accounts.sh
[liam@centos7-vm scripts]$ sudo ./add_test_accounts.sh
Changing password for user fredj.
passwd: all authentication tokens updated successfully.
Changing password for user sallys.
passwd: all authentication tokens updated successfully.
Changing password for user archieb.
passwd: all authentication tokens updated successfully.
Changing password for user earlp.
passwd: all authentication tokens updated successfully.
Changing password for user bertl.
passwd: all authentication tokens updated successfully.
[liam@centos7-vm scripts]$ ls -ltr /home
total 0
drwx------. 5 liam    liam    234 Oct  4 04:28 liam
drwx------. 2 fredj   fredj    81 Oct  4 04:30 fredj
drwx------. 2 sallys  sallys   81 Oct  4 04:30 sallys
drwx------. 2 archieb archieb  81 Oct  4 04:30 archieb
drwx------. 2 earlp   earlp    81 Oct  4 04:30 earlp
drwx------. 2 bertl   bertl    81 Oct  4 04:30 bertl
```

```
$ tail -n 5 /etc/passwd
```

```
[liam@centos7-vm scripts]$ tail -n 5 /etc/passwd
fredj:x:1001:1001::/home/fredj:/bin/bash
sallys:x:1002:1002::/home/sallys:/bin/bash
archieb:x:1003:1003::/home/archieb:/bin/bash
earlp:x:1004:1004::/home/earlp:/bin/bash
bertl:x:1005:1005::/home/bertl:/bin/bash
```

Create Script

In my **scripts** directory, I created an empty script file named **disable_local_user.sh**

```
/home/liam/scripts/disable_local_user.sh
```

I will begin by adding the full path of the command interpreter (in this case **/bin/bash**), as well as, a brief description of what the script does:

```
#!/bin/bash
#
# This script disables, deletes, and/or archives users on the local system.
```

Ensure Superuser Privileges

Then, I want to ensure that the script is being executed by a user with superuser privileges (either **root** or using **sudo**). I will check the environment variable **UID** which is a unique identifier assigned to each user. If the **$UID == 0**, then either the **root** user, or a user with **sudo** privileges, has executed the script.

I will verify the **UID** values assigned to the root user & my non-root user on my running CentOS 7 VM:

```
$ id -u root
$ id -u liam
```

```
[liam@centos7-vm ~]$
[liam@centos7-vm ~]$ id -u root
0
[liam@centos7-vm ~]$ id -u liam
1000
[liam@centos7-vm ~]$
```

We can see from the above command output that the root user has a UID of 0 and my non-root user has a UID of 1000.

Using this code snippet, I am checking if the **root** user, or a non-root user using **sudo** privileges, is executing the script. If not, provide a helpful usage message and exit the script.

```
# Make sure the script is being executed with superuser privileges.
if [[ "${UID}" -ne 0 ]]
then
        echo "You must use superuser privileges to run this script."
        exit 1
fi
```

I will now test this out, for success & failure, by setting the execute permission on the script and executing it:

```
$ chmod +x disable_local_user.sh
$ sudo ./disable_local_user.sh
$ ./disable_local_user.sh
```

```
[liam@centos7-vm scripts]$
[liam@centos7-vm scripts]$ chmod +x disable_local_user.sh
[liam@centos7-vm scripts]$ sudo ./disable_local_user.sh
[liam@centos7-vm scripts]$ ./disable_local_user.sh
You must use superuser privileges to run this script.
[liam@centos7-vm scripts]$
```

You will notice that executing the script without superuser privileges generated the helpful usage message and exited the script.

Script Options using getopts

I will use the bash built-in **getopts** to handle the script options.

```
[liam@centos7-vm scripts]$ help getopts | head -n 5
getopts: getopts optstring name [arg]
    Parse option arguments.

    Getopts is used by shell procedures to parse positional parameters
    as options.
```

Three options can be used when executing this script:

-d Deletes accounts instead of disabling them.
-r Removes the home directory associated with the account(s).
-a Creates an archive of the home directory associated with the account(s).

```
# Parse the options.
while getopts dra OPTION
do
  case ${OPTION} in
    d)
       DELETE_USER='true'
       echo "Delete user."
       ;;
    r)
      REMOVE_HOME='-r'
      echo "Delete user's home directory."
      ;;
    a)
      ARCHIVE='true'
      echo "Archive user's home directory."
      ;;
    ?)
      echo "Invalid option." >&2
      exit 1
      ;;
  esac
done
```

I will now test this out, for success & failure, with the following:

```
$ sudo ./disable_local_user.sh -d
$ sudo ./disable_local_user.sh -r
$ sudo ./disable_local_user.sh -a
$ sudo ./disable_local_user.sh -x
```

```
[liam@centos7-vm scripts]$ sudo ./disable_local_user.sh -d
Delete user.
[liam@centos7-vm scripts]$ sudo ./disable_local_user.sh -r
Delete user's home directory.
[liam@centos7-vm scripts]$ sudo ./disable_local_user.sh -a
Archive user's home directory.
[liam@centos7-vm scripts]$ sudo ./disable_local_user.sh -x
./disable_local_user.sh: illegal option -- x
Invalid option.
```

After all of the script options have been handled by **getopts**, getopts sets the value of **OPTIND** to the position, or index, of where the arguments begin after the script options. Since I need to be able to access usernames specified after the script options, I will use the bash built-in **shift** command to shift the positional parameters to the left. This will remove the script options while leaving the remaining arguments. The remaining arguments will be one, or more, usernames.

```
# Remove the options while leaving the remaining arguments.
shift "$(( OPTIND - 1 ))"

# Ensure at least 1 username is supplied after the options
if [[ "${#}" -lt 1 ]]
then
  echo "You must supply at least 1 valid username." >&2
  exit 1
fi
```

I will now test this out, for success & failure, with the following:

```
$ sudo ./disable_local_user.sh -d earlp sallyj fredf
$ sudo ./disable_local_user.sh -dra jackr
$ sudo ./disable_local_user.sh archieb
$ sudo ./disable_local_user.sh
```

```
[liam@centos7-vm scripts]$ sudo ./disable_local_user.sh -d earlp sallyj fredf
Delete user.
Processing user: earlp
Processing user: sallyj
Processing user: fredf
[liam@centos7-vm scripts]$ sudo ./disable_local_user.sh -dra jackr
Delete user.
Delete user's home directory.
Archive user's home directory.
Processing user: jackr
[liam@centos7-vm scripts]$ sudo ./disable_local_user.sh archieb
Processing user: archieb
[liam@centos7-vm scripts]$ sudo ./disable_local_user.sh
You must supply at least 1 valid username.
```

Script Argument

I want to ensure that at least one username is provided during execution. I will now update the last code snippet to provide helpful usage messages to the user.

```
if [[ "${#}" -lt 1 ]]
then
  # Display the usage and exit.
  echo "Usage: ${0} [-dra] USER [USERN]..." >&2
  echo 'Disable a local Linux account.' >&2
  echo '  -d  Deletes accounts instead of disabling them.' >&2
  echo '  -r  Removes the home directory associated with the account(s).' >&2
  echo '  -a  Creates an archive of the home directory associated with the
account(s).' >&2
  exit 1
fi
```

Since this group of messages could be used more than once, I will create a function.

Usage Function

```
usage() {
  # Display the usage and exit.
  echo "Usage: ${0} [-dra] USER [USERN]..." >&2
  echo 'Disable a local Linux account.' >&2
  echo '  -d  Deletes accounts instead of disabling them.' >&2
  echo '  -r  Removes the home directory associated with the account(s).' >&2
  echo '  -a  Creates an archive of the home directory associated with the
account(s).' >&2
  exit 1
}
```

I can now modify the code snippet that checks whether at least one username was provided.

```
if [[ "${#}" -lt 1 ]]
then
  usage
fi
```

I can also clean up my **getopts** while loop.

```
# Parse the options.
while getopts dra OPTION
do
  case ${OPTION} in
    d) DELETE_USER='true' ;;
    r) REMOVE_HOME='-r' ;;
    a) ARCHIVE='true' ;;
    ?) usage ;;
  esac
done
```

I will now test this out, for success & failure, with the following:

```
$ sudo ./disable_local_user.sh -dra earlp fredj sallys
$ sudo ./disable_local_user.sh archieb
$ sudo ./disable_local_user.sh -x bertl
$ sudo ./disable_local_user.sh
```

```
[liam@centos7-vm scripts]$ sudo ./disable_local_user.sh -dra earlp fredj sallys
Processing user: earlp
Processing user: fredj
Processing user: sallys
[liam@centos7-vm scripts]$ sudo ./disable_local_user.sh archieb
Processing user: archieb
[liam@centos7-vm scripts]$ sudo ./disable_local_user.sh -x bertl
./disable_local_user.sh: illegal option -- x
Usage: ./disable_local_user.sh [-dra] USER [USERN]...
Disable a local Linux account.
  -d  Deletes accounts instead of disabling them.
  -r  Removes the home directory associated with the account(s).
  -a  Creates an archive of the home directory associated with the account(s).
[liam@centos7-vm scripts]$ sudo ./disable_local_user.sh
Usage: ./disable_local_user.sh [-dra] USER [USERN]...
Disable a local Linux account.
  -d  Deletes accounts instead of disabling them.
  -r  Removes the home directory associated with the account(s).
  -a  Creates an archive of the home directory associated with the account(s).
```

check_status Function

Before proceeding, I will create a function that will be used to check the status of executed commands. This will prevent me from having to repeat myself throughout the script.

```
check_status(){
  MESSAGE="${1}"
  if [[ "${?}" -ne 0 ]]
  then
    echo "${MESSAGE} FAILED" >&2
    exit 1
  fi
}
```

Account Operations

After having shifted the positional parameters to the left, the options were removed and I am left with one, or more, usernames. For each username passed, I will first check that they exist, and then ensure that their **UID >= 1000** (not a system account).

```
# Loop through all the usernames supplied as arguments.
for USERNAME in "${@}"
do
  echo "Processing user: ${USERNAME}"

  # array to store account operations performed
  OPERATIONS=''

  # ensure user exists
  if ! id ${USERNAME} &> /dev/null
  then
       echo "${USERNAME} does NOT exist." >&2
       exit 1
  fi

  USERID=$(id -u ${USERNAME})
  if [[ "${USERID}" -lt 1000 ]]
  then
    echo "Refusing to remove the ${USERNAME} account with UID ${USERID}." >&2
    echo "You can only disable, delete and/or archive user accounts." >&2
    exit 1
  fi
done
```

I will now test this out, for success & failure, with the following:

```
$ sudo ./disable_local_user.sh -dra archieb
$ sudo ./disable_local_user.sh -d larryh
$ sudo ./disable_local_user.sh ftp
```

```
[liam@centos7-vm scripts]$ sudo ./disable_local_user.sh -dra archieb
Processing user: archieb
[liam@centos7-vm scripts]$ sudo ./disable_local_user.sh -d larryh
Processing user: larryh
larryh does NOT exist.
[liam@centos7-vm scripts]$ sudo ./disable_local_user.sh ftp
Processing user: ftp
Refusing to remove the ftp account with UID 14.
You can only disable, delete and/or archive user accounts.
```

Archive Account

Still in the **for** loop, I will now check if the archive (**-a**) option was specified during script execution. First, I will specify where to archive user accounts.

```
  ARCHIVE_DIR='/archive/users'

  # Create an archive if requested to do so.
  if [[ "${ARCHIVE}" = 'true' ]]
  then
    # Make sure the ARCHIVE_DIR directory exists.
    if [[ ! -d "${ARCHIVE_DIR}" ]]
    then
      echo "Creating ${ARCHIVE_DIR} directory."
      mkdir -p ${ARCHIVE_DIR}
      check_status "Creating ${ARCHIVE_DIR} directory "
    fi

    # Archive the user's home directory and move it into the ARCHIVE_DIR
    HOME_DIR="/home/${USERNAME}"
    ARCHIVE_FILE="${ARCHIVE_DIR}/${USERNAME}.tgz"
    if [[ -d "${HOME_DIR}" ]]
    then
      echo "Archiving ${HOME_DIR} to ${ARCHIVE_FILE}"
      tar -zcf ${ARCHIVE_FILE} ${HOME_DIR} &> /dev/null
      check_status "Archiving ${HOME_DIR} to ${ARCHIVE_FILE}"

      # add to array of account operations performed
      OPERATIONS+=" archived"
    else
      echo "${HOME_DIR} does not exist or is not a directory." >&2
      exit 1
    fi
  fi # END of if "${ARCHIVE}" = 'true'
```

Delete/Disable Account

Next, still in the **for** loop, I will check if user is to be deleted **(-d**) and whether, or not, to delete their home directory (**-r**).

```
  if [[ "${DELETE_USER}" = 'true' ]]
  then
    # Delete the user.
    userdel ${REMOVE_HOME} ${USERNAME}
    check_status "Deleting account ${USERNAME}"

    # add to array of account operations performed
    if [[ -z ${REMOVE_HOME} ]]
    then
      OPERATIONS+=' deleted'
    else
       OPERATIONS+=' deleted and home directory removed'
    fi
  else
```

```
    # Disable the user.
    chage -E 0 ${USERNAME}
    check_status "Disabling account ${USERNAME}"

    # add to array of account operations performed
    OPERATIONS=" disabled"
  fi # END of if "${DELETE_USER}" = 'true'
```

Display Script Output

After having performed all the required operations, I can now display the username along with any actions performed against their account. Finally, I can close out the outer loop that dealt with the remaining script arguments (username(s)).

```
  # Display the username and any actions performed against the account.
  echo "${USERNAME}" was:
  for OP in "${OPERATIONS}"
  do
      echo -e "\t${OP}"
  done
done
exit 0
```

Review Script

```
#!/bin/bash
#
# This script disables, deletes, and/or archives users on the local system.

ARCHIVE_DIR='/archive/users'

usage() {
  # Display the usage and exit.
  echo "Usage: ${0} [-dra] USER [USERN]..." >&2
  echo 'Disable a local Linux account.' >&2
  echo '  -d  Deletes accounts instead of disabling them.' >&2
  echo '  -r  Removes the home directory associated with the account(s).' >&2
  echo '  -a  Creates an archive of the home directory associated with the account(s).' >&2
  exit 1
}

check_status(){
  MESSAGE="${1}"
  if [[ "${?}" -ne 0 ]]
  then
    echo "${MESSAGE} FAILED" >&2
    exit 1
  fi
}

# Make sure the script is being executed with superuser privileges.
if [[ "${UID}" -ne 0 ]]
then
   echo "You must use superuser privileges to run this script." >&2
   exit 1
fi
```

```
# Parse the options.
while getopts dra OPTION
do
  case ${OPTION} in
    d) DELETE_USER='true' ;;
    r) REMOVE_HOME='-r' ;;
    a) ARCHIVE='true' ;;
    ?) usage ;;
  esac
done

# Remove the options while leaving the remaining arguments.
shift "$(( OPTIND - 1 ))"

# If the user doesn't supply at least one argument, provide help.
if [[ "${#}" -lt 1 ]]
then
  usage
fi

# Loop through all the usernames supplied as arguments.
for USERNAME in "${@}"
do
  echo "Processing user: ${USERNAME}"

  # array to store account operations performed
  OPERATIONS=''

  # ensure user exists
  if ! id ${USERNAME} &> /dev/null
  then
    echo "${USERNAME} does NOT exist." >&2
    exit 1
  fi

  # Make sure the UID of the account is at least 1000.
  USERID=$(id -u ${USERNAME})
  if [[ "${USERID}" -lt 1000 ]]
  then
    echo "Refusing to remove the ${USERNAME} account with UID ${USERID}." >&2
    echo "You can only disable, delete and/or archive user accounts." >&2
    exit 1
  fi

  # Create an archive if requested to do so.
  if [[ "${ARCHIVE}" = 'true' ]]
  then
    # Make sure the ARCHIVE_DIR directory exists.
    if [[ ! -d "${ARCHIVE_DIR}" ]]
    then
      echo "Creating ${ARCHIVE_DIR} directory."
      mkdir -p ${ARCHIVE_DIR}
      check_status "Creating ${ARCHIVE_DIR} directory "
    fi

    # Archive the user's home directory and move it into the ARCHIVE_DIR
    HOME_DIR="/home/${USERNAME}"
```

```
    ARCHIVE_FILE="${ARCHIVE_DIR}/${USERNAME}.tgz"
    if [[ -d "${HOME_DIR}" ]]
    then
      echo "Archiving ${HOME_DIR} to ${ARCHIVE_FILE}"
      tar -zcf ${ARCHIVE_FILE} ${HOME_DIR} &> /dev/null
      check_status "Archiving ${HOME_DIR} to ${ARCHIVE_FILE}"

      # add to array of account operations performed
      OPERATIONS+=" archived"
    else
      echo "${HOME_DIR} does not exist or is not a directory." >&2
      exit 1
    fi
  fi # END of if "${ARCHIVE}" = 'true'

  if [[ "${DELETE_USER}" = 'true' ]]
  then
    # Delete the user.
    userdel ${REMOVE_HOME} ${USERNAME}
    check_status "Deleting account ${USERNAME}"

    # add to array of account operations performed
    if [[ -z ${REMOVE_HOME} ]]
    then
      OPERATIONS+=' deleted'
    else
      OPERATIONS+=' deleted and home directory removed'
    fi
  else
    # Disable the user.
    chage -E 0 ${USERNAME}
    check_status "Disabling account ${USERNAME}"
    # add to array of account operations performed
    OPERATIONS=" disabled"
  fi # END of if "${DELETE_USER}" = 'true'

  # Display the username and any actions performed against the account.
  echo "${USERNAME}" was:
  for OP in "${OPERATIONS}"
  do
       echo -e "\t${OP}"
  done
done

exit 0
```

Test Script

Before testing the script, I will first confirm that the user home directories exist.

```
[liam@centos7-vm scripts]$ ls -ltr /home
total 0
drwx------.  5 liam     liam     234 Oct   4 04:58 liam
drwx------.  2 sallys   sallys    62 Oct   4 06:09 sallys
drwx------.  2 archieb  archieb   62 Oct   4 06:09 archieb
drwx------.  2 fredj    fredj     62 Oct   4 06:09 fredj
drwx------.  2 bertl    bertl     84 Oct   4 06:22 bertl
drwx------.  2 earlp    earlp     84 Oct   4 06:22 earlp
```

Then, I will create a simple file for two accounts that will be archived during testing. This will enable me to validate the compressed archive by checking its contents.

```
$ sudo sh -c 'echo "This is the home directory of Bert Lomax" > /home/bertl/bert_lomax.txt'
$ sudo sh -c 'echo "This is the home directory of Earl Pearl" > /home/earlp/earl_pearl.txt'
$ sudo cat /home/bertl/bert_lomax.txt
$ sudo cat /home/earlp/earl_pearl.txt
```

```
[liam@centos7-vm scripts]$
[liam@centos7-vm scripts]$ sudo sh -c 'echo "This is the home directory of Bert Lomax"
> /home/bertl/bert_lomax.txt'
[liam@centos7-vm scripts]$
[liam@centos7-vm scripts]$ sudo sh -c 'echo "This is the home directory of Earl Pearl"
> /home/earlp/earl_pearl.txt'
[liam@centos7-vm scripts]$
[liam@centos7-vm scripts]$ sudo cat /home/bertl/bert_lomax.txt
This is the home directory of Bert Lomax
[liam@centos7-vm scripts]$
[liam@centos7-vm scripts]$ sudo cat /home/earlp/earl_pearl.txt
This is the home directory of Earl Pearl
[liam@centos7-vm scripts]$
```

I will first test the script for failure with the following:

```
$ ./disable_local_user.sh
$ sudo ./disable_local_user.sh
```

```
[liam@centos7-vm scripts]$ ./disable_local_user.sh
You must use superuser privileges to run this script.
[liam@centos7-vm scripts]$
[liam@centos7-vm scripts]$ sudo ./disable_local_user.sh
Usage: ./disable_local_user.sh [-dra] USER [USERN]...
Disable a local Linux account.
  -d  Deletes accounts instead of disabling them.
  -r  Removes the home directory associated with the account(s).
  -a  Creates an archive of the home directory associated with the account(s).
```

```
$ sudo ./disable_local_user.sh -x bertl
$ sudo ./disable_local_user.sh ftp
```

```
[liam@centos7-vm scripts]$ sudo ./disable_local_user.sh -x bertl
./disable_local_user.sh: illegal option -- x
Usage: ./disable_local_user.sh [-dra] USER [USERN]...
Disable a local Linux account.
  -d  Deletes accounts instead of disabling them.
  -r  Removes the home directory associated with the account(s).
  -a  Creates an archive of the home directory associated with the account(s).
[liam@centos7-vm scripts]$
[liam@centos7-vm scripts]$ sudo ./disable_local_user.sh ftp
Processing user: ftp
Refusing to remove the ftp account with UID 14.
You can only disable, delete and/or archive user accounts.
```

I will now test the script for success with the following:

$ sudo ./disable_local_user.sh fredj

```
[liam@centos7-vm scripts]$ sudo ./disable_local_user.sh fredj
Processing user: fredj
fredj was:
        disabled
```

I can verify that the account was disabled by trying to login as **fredj**.

$ su - fredj

```
[liam@centos7-vm scripts]$ su - fredj
Password:
Your account has expired; please contact your system administrator
su: User account has expired
```

$ sudo ./disable_local_user.sh -d archieb

```
[liam@centos7-vm scripts]$ sudo ./disable_local_user.sh -d archieb
Processing user: archieb
archieb was:
        deleted
```

I can verify that the account was deleted by trying to login as **archieb**.

$ su - archieb

```
[liam@centos7-vm scripts]$
[liam@centos7-vm scripts]$ su - archieb
su: user archieb does not exist
[liam@centos7-vm scripts]$
```

I can also confirm that the home directory was not deleted.

$ ls -l /home

```
[liam@centos7-vm scripts]$ ls -l /home
total 0
drwx------. 2   1003   1003  62 Oct  4 06:09 archieb
drwx------. 2 bertl  bertl   84 Oct  4 06:22 bertl
drwx------. 2 earlp  earlp   84 Oct  4 06:22 earlp
drwx------. 2 fredj  fredj   62 Oct  4 06:09 fredj
drwx------. 5 liam   liam   234 Oct  4 07:00 liam
drwx------. 2 sallys sallys  62 Oct  4 06:09 sallys
```

$ sudo ./disable_local_user.sh -dr sallys

```
[liam@centos7-vm scripts]$ sudo ./disable_local_user.sh -dr sallys
Processing user: sallys
sallys was:
        deleted and home directory removed
```

$ su - sallys

$ sudo ls -l /home/sallys

```
[liam@centos7-vm scripts]$
[liam@centos7-vm scripts]$ su - sallys
su: user sallys does not exist
[liam@centos7-vm scripts]$
[liam@centos7-vm scripts]$ sudo ls -l /home/sallys
ls: cannot access /home/sallys: No such file or directory
[liam@centos7-vm scripts]$
```

$ sudo ./disable_local_user.sh -dra bertl earlp

```
[liam@centos7-vm scripts]$
[liam@centos7-vm scripts]$ sudo ./disable_local_user.sh -dra bertl earlp
Processing user: bertl
Creating /archive/users directory.
Archiving /home/bertl to /archive/users/bertl.tgz
bertl was:
        archived deleted and home directory removed
Processing user: earlp
Archiving /home/earlp to /archive/users/earlp.tgz
earlp was:
        archived deleted and home directory removed
[liam@centos7-vm scripts]$
```

$ su - bertl
$ su - earlp

```
[liam@centos7-vm scripts]$
[liam@centos7-vm scripts]$ su - bertl
su: user bertl does not exist
[liam@centos7-vm scripts]$
[liam@centos7-vm scripts]$ su - earlp
su: user earlp does not exist
[liam@centos7-vm scripts]$
```

Now I can confirm that their home directories were deleted.

$ sudo ls -l /home/bertl
$ sudo ls -l /home/earlp

```
[liam@centos7-vm scripts]$
[liam@centos7-vm scripts]$ sudo ls -l /home/bertl
ls: cannot access /home/bertl: No such file or directory
[liam@centos7-vm scripts]$
[liam@centos7-vm scripts]$ sudo ls -l /home/earlp
ls: cannot access /home/earlp: No such file or directory
[liam@centos7-vm scripts]$
```

I can also ensure that the home directories of both users, **bertl** & **earlp**, were archived.

```
$ sudo ls -l /archive/users
```

```
[liam@centos7-vm scripts]$
[liam@centos7-vm scripts]$ ls -l /archive/users
total 8
-rw-r--r--. 1 root root 551 Oct  4 07:13 bertl.tgz
-rw-r--r--. 1 root root 547 Oct  4 07:13 earlp.tgz
[liam@centos7-vm scripts]$
```

Finally, I check the contents of each archive to ensure that the simple file I added pre-testing is part of the archive.

```
$ sudo tar -tvf /archive/users/bertl.tgz
$ sudo tar -tvf /archive/users/earlp.tgz
```

```
[liam@centos7-vm scripts]$
[liam@centos7-vm scripts]$ sudo tar -tvf /archive/users/bertl.tgz
drwx------ bertl/bertl       0 2022-10-04 06:22 home/bertl/
-rw-r--r-- bertl/bertl      18 2021-11-24 11:33 home/bertl/.bash_logout
-rw-r--r-- bertl/bertl     193 2021-11-24 11:33 home/bertl/.bash_profile
-rw-r--r-- bertl/bertl     231 2021-11-24 11:33 home/bertl/.bashrc
-rw-r--r-- root/root        41 2022-10-04 06:22 home/bertl/bert_lomax.txt
[liam@centos7-vm scripts]$
[liam@centos7-vm scripts]$ sudo tar -tvf /archive/users/earlp.tgz
drwx------ earlp/earlp       0 2022-10-04 06:22 home/earlp/
-rw-r--r-- earlp/earlp      18 2021-11-24 11:33 home/earlp/.bash_logout
-rw-r--r-- earlp/earlp     193 2021-11-24 11:33 home/earlp/.bash_profile
-rw-r--r-- earlp/earlp     231 2021-11-24 11:33 home/earlp/.bashrc
-rw-r--r-- root/root        41 2022-10-04 06:22 home/earlp/earl_pearl.txt
[liam@centos7-vm scripts]$
```

I hope you have enjoyed completing this tutorial and found it helpful.

If you would like to know how to create a shell script that can be used to create Linux user accounts with auto-generated secure passwords, my tutorial **Create User Accounts** is accessible **here**. My Linux tutorials page is **here**, while my main tutorials page is accessible **here**.

Back to Top