

Day 3: What is a directory or file?

- Unix/linux sees “everything” in its system as a file of some type
- Both dirs and files are reserved “areas” in a “file system” - in memory or a hard drive, for example - that has an amount of available “space” allocated for it's use, that may or may not contain data. (Empty dir or file)
- A file has to have a directory to define it. Why?
- **delete/rename/recreate is controlled by the parent directory permissions, because that is where the name to data space mapping is stored.**
- **This means the owner of a directory containing another user's file may be able to delete it, even though they don't own it, or can't write to it, depending on how the permissions are set on either.**
- **SO, file protection has to be considered by the write bits on the container, and who and what group/others can access the directory - BEWARE 777 on a dir! Everyone may be able to RWX the contents!!**

File and Directory Permissions

Default creation permissions:

```
stevee@MintServer ~ $ mkdir TestDir
```

```
stevee@MintServer ~ $ touch TestDir/testfile.txt
```

```
stevee@MintServer ~ $ ls -al TestDir/
```

```
total 8
```

```
drwxr-xr-x 2 stevee stevee 4096 Sep 22 10:50 .
```

```
drwxr-xr-x 37 stevee stevee 4096 Sep 22 10:49 ..
```

```
-rw-r--r-- 1 stevee stevee    0 Sep 22 10:50
```

```
testfile.txt
```

Dir = 755

File = 644

How and Why? What do the numbers mean?

What does rwx mean?

Read, write and execute permission
Applied to a directory and a file

Read = 4

Write = 2

Exe = 1

All add up to 7 max (i.e. octal), for each UGO block. e.g.

$r+w = 6 \ (4+2) = (110 \text{ binary})$

$r+x = 5 \ (4+1) = (101 \text{ binary})$

$r+w+x = 7 \ (4+2+1) = (111 \text{ binary})$

$--- = 0 \ (0+0+0) = (000 \text{ binary})$

You can use numbers or text to change permissions on a dir or file. Using numbers requires all 3 blocks, but text can change just ONE block – more convenient:

`chmod 777 /file.txt = chmod ugo+rwx /file.txt`

Can user joe delete user stevee's file?

- stevee@MintServer ~ \$ **su joe**
- Password:
- joe@MintServer /home/stevee \$ **rm -v TestDir/testfile.txt**
- **rm: remove write-protected regular empty file 'TestDir/testfile.txt'? y**
- **rm: cannot remove 'TestDir/testfile.txt': Permission denied**
- No! Joe cannot delete stevee's file from stevee's directory
- Why? Joe does not have write permissions to stevee's **dir**, and is not in stevee's group – even if group write permissions were set – which they aren't!
- **ls -ld TestDir/**
- **drwxr-xr-x 2 stevee stevee 4096 Sep 22 11:26 TestDir/**

How can user joe delete user stevee's file if required?

- Group write permissions have to be added to stevee's **directory** – remember – this holds the file definitions
- Joe needs to be in stevee's group
- stevee@MintServer ~ \$ `sudo adduser joe stevee`
- Adding user `joe' to group `stevee' ...
- Adding user joe to group stevee
- stevee@MintServer ~ \$ `chmod g+w Testdir/`
- stevee@MintServer ~ \$ `ls -ld Testdir/`
- `drwxrwxr-x 2 stevee stevee 4096 Sep 22 12:21 Testdir/`
- stevee@MintServer ~ \$ `su joe`
- Password:
- joe@MintServer /home/stevee \$ `rm -vr Testdir/`
- `rm: remove write-protected regular empty file 'Testdir/tesfile.txt'? y`
- **removed 'Testdir/tesfile.txt'**
- `rm: cannot remove 'Testdir/': Permission denied`
- Joe **can delete the file** but NOT the directory, as he does not OWN it, but has write permissions to its contents!
- Group membership affects directory contents here, not the directory itself

Can user root delete user stevee's dir?

- MintServer stevee # `rmkdir -v TestDir/`
- `rmkdir: removing directory, 'TestDir/'`
- `rmkdir: failed to remove 'TestDir/': Directory not empty`
- MintServer stevee # `ls -al TestDir/testfile.txt`
- `-rw-r--r-- 1 stevee stevee 0 Sep 22 10:58 TestDir/testfile.txt`
- **It depends on the command used:**

Can user root delete user stevee's file?

- stevee@MintServer ~ \$ **su root**
- Password:
- MintServer stevee # **rm -vr TestDir/testfile.txt**
- **removed 'TestDir/testfile.txt'**
- Yes! Root user can delete stevee's file from stevee's directory.
- **Why? He is not in stevee's group?**
- (he wouldn't be much use as Superuser else..!).

Can user stevee delete root's file?

- stevee@MintServer ~ \$ /home/stevee \$ **su**
- **Password:**
- MintServer stevee # **touch TestDir/rootsfile.txt**
- MintServer stevee # **ls -l TestDir/**
- **-rw-r--r-- 1 root root 0 Sep 22 11:23 rootsfile.txt**
- **-rw-r--r-- 1 stevee stevee 0 Sep 22 10:58 testfile.txt**
- stevee@MintServer ~ \$ **rm -v TestDir/rootsfile.txt**
- **rm: remove write-protected regular empty file 'TestDir/rootsfile.txt'? y**
- **removed 'TestDir/rootsfile.txt'**
- Yes! Steve can remove root's file from stevee's dir even though he has;
- no write permissions to root's file;
- is NOT in root' group
- He DOES have write perms to his own directory.
- stevee@MintServer ~ \$ **ls -ld TestDir/**
- **drwxr-xr-x 2 stevee stevee 4096 Sep 22 11:26 TestDir/**

Can a user create a file in any of root's directories?

Depends on it's perms!

First, create a dir as root.

```
stevee@AMDA8 ~ $ su root
```

Password:

```
AMDA8 stevee # mkdir RootsDir
```

```
AMDA8 stevee # ls -al RootsDir/
```

```
total 8
```

```
drwxr-xr-x  2 root  root  4096 Sep 20 19:52 .
```

```
drwxr-xr-x 26 stevee stevee 4096 Sep 20 19:52 ..
```

Now, as user stevee, create a file in RootsDir

```
AMDA8 stevee # exit
```

```
stevee@AMDA8 ~ $ touch RootsDir/user.txt
```

```
touch: cannot touch 'RootsDir/user.txt': Permission denied
```

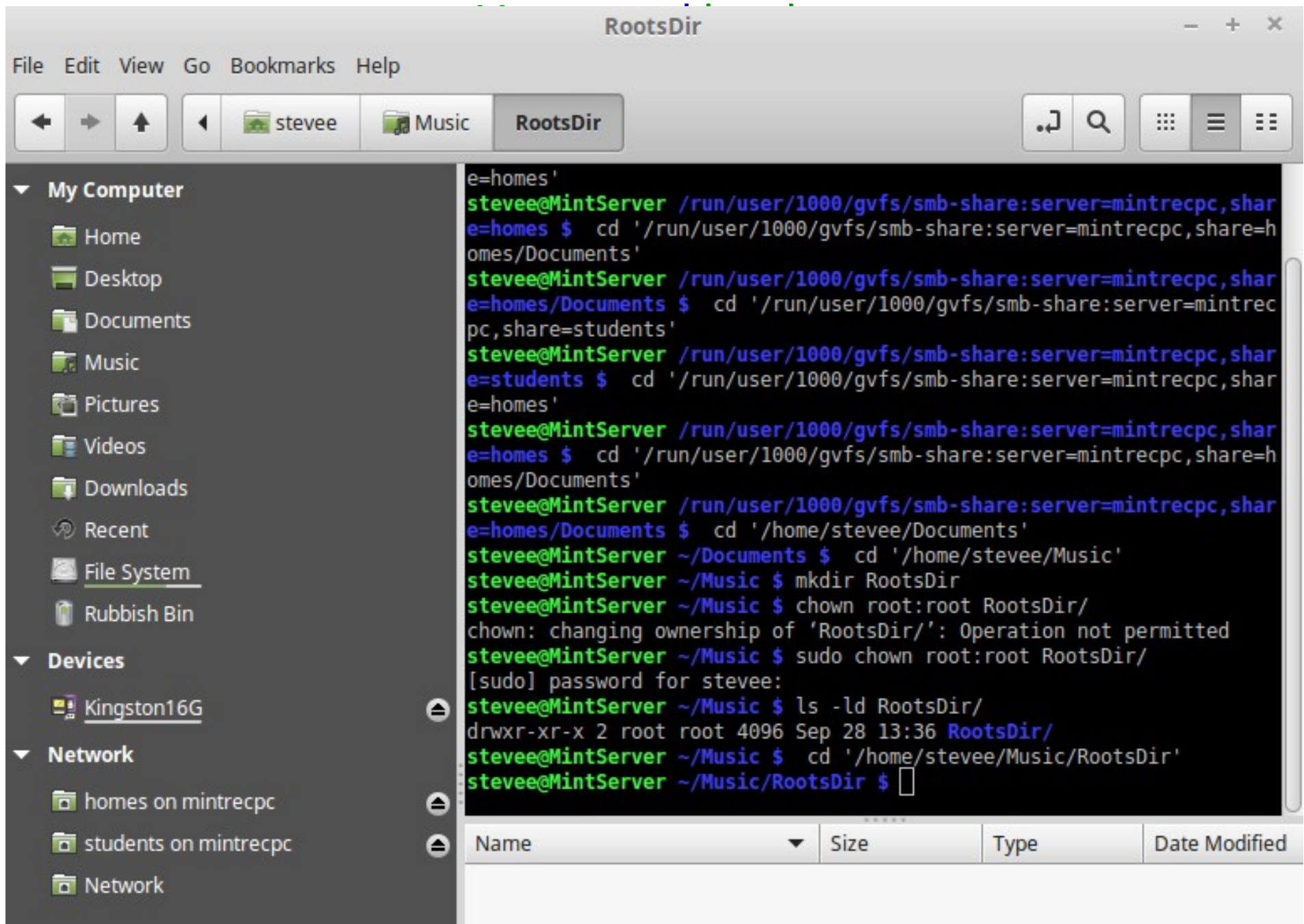
Users cannot create or delete files in root's *default perms* folders – for obvious reasons – trojans etc.

Open RootsDir in the Desktop view:

RootsDir Traversal – allowed

Why? What perms exist?

Click on RootsDir in Nemo File Manager



What relation is user to root in terms of UGO?

- Stevee is not root
- Stevee is not in root's group
- What's left?
- Others! Look at the RootsDir perms:
- AMDA8 stevee # `ls -al RootsDir/`
- `drwxr-xr-x 2 root root 4096 Sep 20 19:52 .`
- Look in your RootsDir in the Desktop – it's empty, and you can move into it in the GUI:

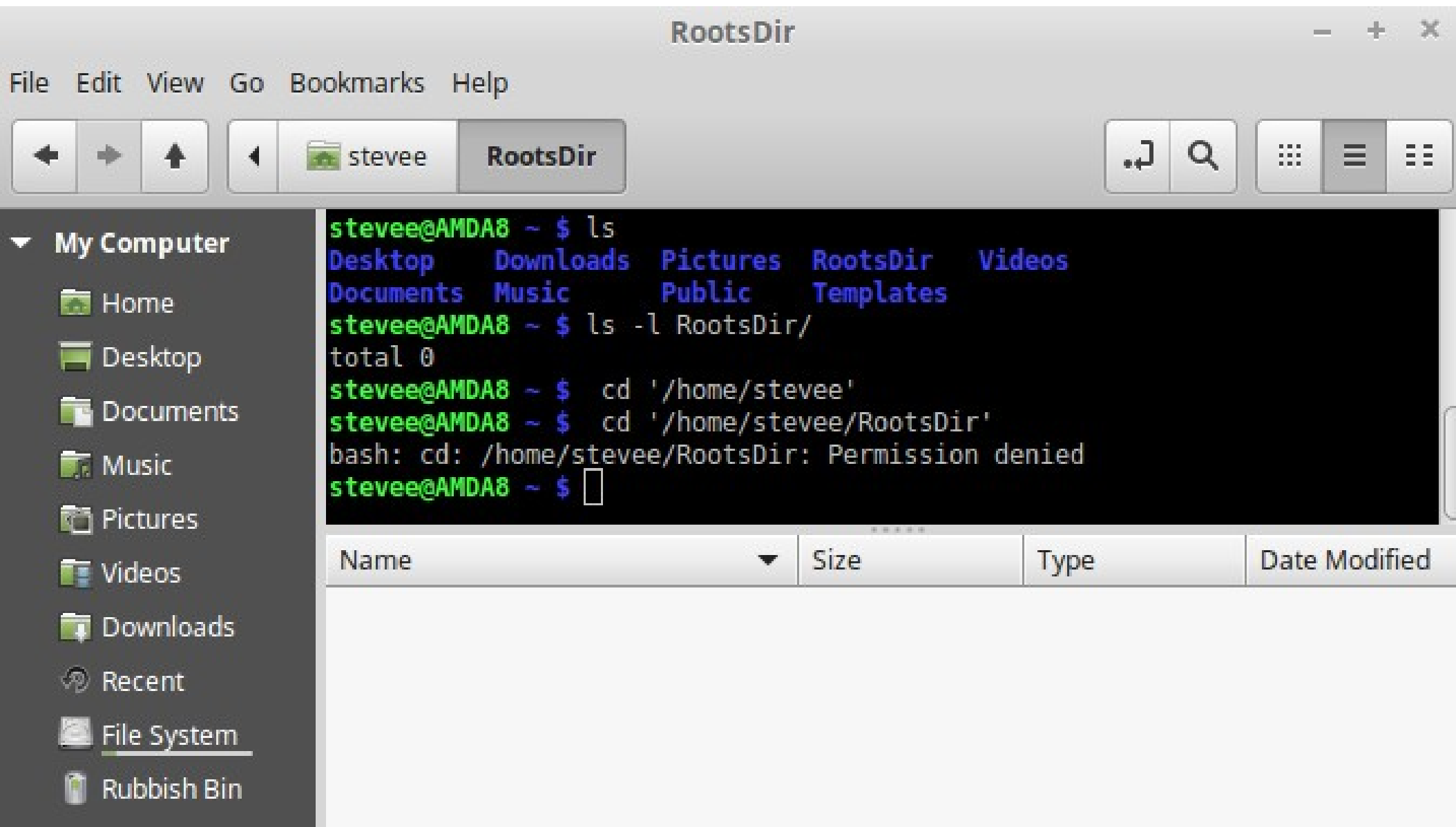
Change RootsDir Perms:

chmod o-x

stevee@AMDA8 ~ \$ su root

- Password:
- AMDA8 stevee # chmod o-x RootsDir/
- AMDA8 stevee # ls -al RootsDir/
- `drwxr-xr-- 2 root root 4096 Sep 20 19:52 .`
- Now look in the Desktop and try to open RootsDir again – you cannot `cd` into it any more without the Other's exe permissions!

RootsDir Access on Desktop



When can anyone delete anyone else's files – even root's? Bill is not in stevee's or root's group, so has no perms on the containing directory
“RootsDir”

```
MintRecPC Music # chmod o+w RootsDir/
```

```
MintRecPC Music # ls -ld RootsDir/
```

```
drwxr-xrwx 2 root root 4096 Sep 25 12:06 RootsDir/
```

```
MintRecPC Music # su bill
```

```
bill@MintRecPC /home/stevee/Music $
```

```
bill@MintRecPC /home/stevee/Music $ rm -vr RootsDir/
```

```
rm: remove write-protected regular empty file 'RootsDir/rootsfile.txt'? y
```

```
removed 'RootsDir/rootsfile.txt'
```

```
rm: cannot remove 'RootsDir/': Permission denied
```

Oops! No rootsfile.txt anymore!

Beware Others with -----rwx perms = --7, on dirs and/or files!!

Everyone may have full write permissions and can delete or run files!

If the containing directory has rx perms only for Others, then at least the user/owner has some protection for the files contained within.

Hence these default creation perms for read and traversal/execute, but NOT WRITE, reasons:

```
ls -al RootsDir/
```

```
drwxr-xr-x 2 root root 4096 Sep 20 21:09 .
```

```
-rw-r--r-- 1 root root 0 Sep 20 21:09 rootfile
```

rm – remove – the most dangerous command ever??

After the last example, if you were root, what do YOU think would happen if you issued:

rm -vr /

???????

Let's read the:

man rm

intro

man rm

rm - remove files or directories

DESCRIPTION

This manual page documents the GNU version of rm. rm removes each specified file. **By default, it does not remove directories.** (Try it for yourself)

If the -I or --interactive=once option is given, and there are more than three files or the -r, -R, or --recursive are given, then rm prompts the user for whether to proceed with the entire operation. If the response is not affirmative, the entire command is aborted.

Otherwise, if a file is unwritable, standard input is a terminal, and the -f or --force option is not given, or the -i or --interactive=always option is given, rm prompts the user for whether to remove the file. If the response is not affirmative, the file is skipped.

Current versions have this failsafe built in – early versions did not!!

rm -r /

deleted the whole filesystem of the OS!!!

chmod examples

AMDA8 Music # `chmod ugo+rwx RootsDir/rootfile`

AMDA8 Music # `ls -l RootsDir/rootfile`

`-rwxrwxrwx` 1 root root 0 Sep 20 21:40
RootsDir/rootfile

AMDA8 Music # `chmod ugo-rwx RootsDir/rootfile`

AMDA8 Music # `ls -l RootsDir/rootfile`

`------` 1 root root 0 Sep 20 21:40 RootsDir/rootfile

AMDA8 Music # `chmod u+rwx,g+rx RootsDir/rootfile`

AMDA8 Music # `ls -l RootsDir/rootfile`

`-rwxr-x---` 1 root root 0 Sep 20 21:40 RootsDir/rootfile

Day Summary - Permissions

- ***File name defined in the containing dir structure***
- Dir write perm affects dir contents deletion
- Default create permissions: dir **755** and file **644**
- **755: Good compromise of access and security**
- R=4 ; W = 2; X =1 to a max of 7 for each UGO
- Root can change access perms on all dirs/files
- Perms **rx** are required to view/enter (traverse) a dir
- **rmdir** and **rm** commands and their differences
- Shared access via group membership and perms
- Users cannot write to any *default* perm root dir (malware)
- Failsafe for rm command (-i, interactive remove y/n)

Extra reading

- Get a good Admin reference book!
- Beware Googling “Linux file permissions”, re no mention of dir perms allowing content deletion – just the assumption that perms affect ONLY the dir/file they list as, with `ls -l`! They don't always!
- Mostly people copying others text without experimentation or good source info like Linux System Admin; Nemeth, Snyder, Hein
- http://linuxcommand.org/lc3_Its0090.php
- Best way to learn is to experiment - change perms, access files as different users, local and remote – things often don't do what you think!