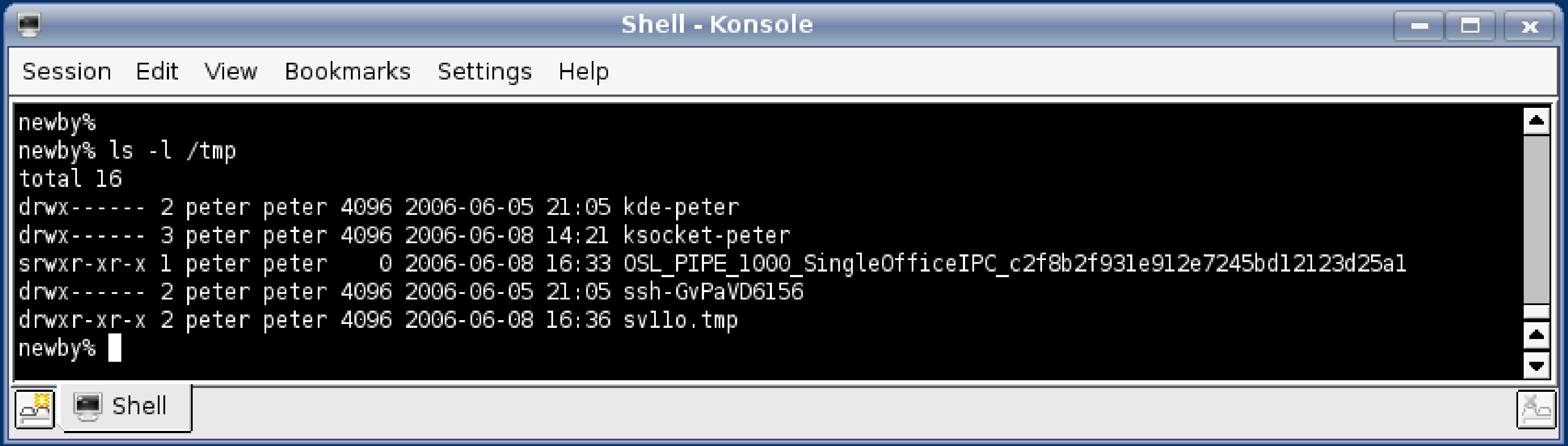
# *File Properties and Permissions*

Managing File Access in Linux

Peter Perry
July 2009

## *What is it about?*



```
newby%
newby% ls -l /tmp
total 16
drwx------ 2 peter peter 4096 2006-06-05 21:05 kde-peter
drwx------ 3 peter peter 4096 2006-06-08 14:21 ksocket-peter
srwxr-xr-x 1 peter peter    0 2006-06-08 16:33 OSL_PIPE_1000_SingleOfficeIPC_c2f8b2f931e912e7245bd12123d25a1
drwx------ 2 peter peter 4096 2006-06-05 21:05 ssh-GvPaVD6156
drwxr-xr-x 2 peter peter 4096 2006-06-08 16:36 sv1lo.tmp
newby%
```

- Open a shell (terminal) and type “ls -l”
- You get quite a bit of information about each file.
- Tonight, we are going to explore some of that information

## *But Why Should I Care?*

- The short answer is that if you are the only user of your computer, you may not have to.
- But how do you stop your grandson from inadvertently doing the equivalent of “rm -rf /”?
- How can you let some users access some of your files, while stopping others?
- And anyway, what does it mean?

## *So, start with something simple*

- Linux, like most flavours of Unix, remembers quite a bit about each file:
  - Who **created** it and when
  - Who last **modified** it and when
  - How **large** it is
  - What **group** it belongs to (more on this later)
  - What type of file it is (directory, link, data file)

## *Have you ever looked at your /etc/passwd file?*

```
games:x:5:60:games:/usr/games:/bin/sh
statd:x:108:65534::/var/lib/nfs:/bin/false
bianka:x:1001:1001:Bianka:/home/bianka:/usr/bin/zsh
work:x:28315:0:Peter:/home/work:/bin/bash
```

- This is a small excerpt from mine.
- It establishes user name, user ID, default group, home directory, and the shell you use.
- It does NOT establish your password (it used to).

## And Groups?

- Groups are defined in the file “/etc/group”
- When you log in, you are in your default group
- Normally, any new file you create will be assigned to that group and be owned by you.

## File Permissions

- In early Unix, it was three octal digits.
- Each digit controlled one part of the access.
  - The first digit is for the owner of the file (you) - u
  - The second digit is for members of the group - g
  - The third digit is for everyone else - o
- Within a digit three bits control types of access
  - 4 – read (r)
  - 2 – write (w)
  - 1 – execute (x)

## *Directory Permissions*

- You must have “execute” permission to use a directory as a directory.
- You must have write permission to create files in a directory.

## *Too hard?*

- These days we do it all symbolically, using the “chmod” command
  - chmod g+x – add execute permissions for the group
  - chmod u-w – stop yourself accidentally deleting it (you will be prompted whether you really want to)
  - chmod a+rx – allow everyone to read and execute it.
- We will come back to what these mean

## So you can set up several accounts on your computer

- Each with its own password
- Each can have exclusive access to their own files (of course, you are the superuser and can do anything)
- Each can share the files they want with other users
- You can make several groups for even more flexibility.

## The commands

- chmod [-mode] [files]
  sets the access permissions for files.
- chgrp [-options] group [files]
  sets the group a file is in (by default your files will usually be assigned to your default group).

## chmod modes

- The mode is [who][+|-|=][category]
- Multiple modes can be given, separated by commas.
- [who] is u (user), g (group), o(other), or a (all).
- [category] is one or more of 'rwxXstugo'
- + means add this permission, - means remove it, and = means set (removing all others).

## chmod modes

- r – read
- w – write
- x – execute (or directory access)
- X – execute/directory access, only if it is a directory or already has execute access.
- s – set id bit (see later)
- t – set sticky bit (forget it)
- u – copy the user permissions
- g – copy the group permissions
- o – copy the other permissions

## *Umask*

- Setting a umask enables you to control the default permissions on a file
- The mask specifies the permission bits that are NOT set. - e.g. umask 022 means owner has all permissions, group and world do not have write permissions.

## Set ID Bits

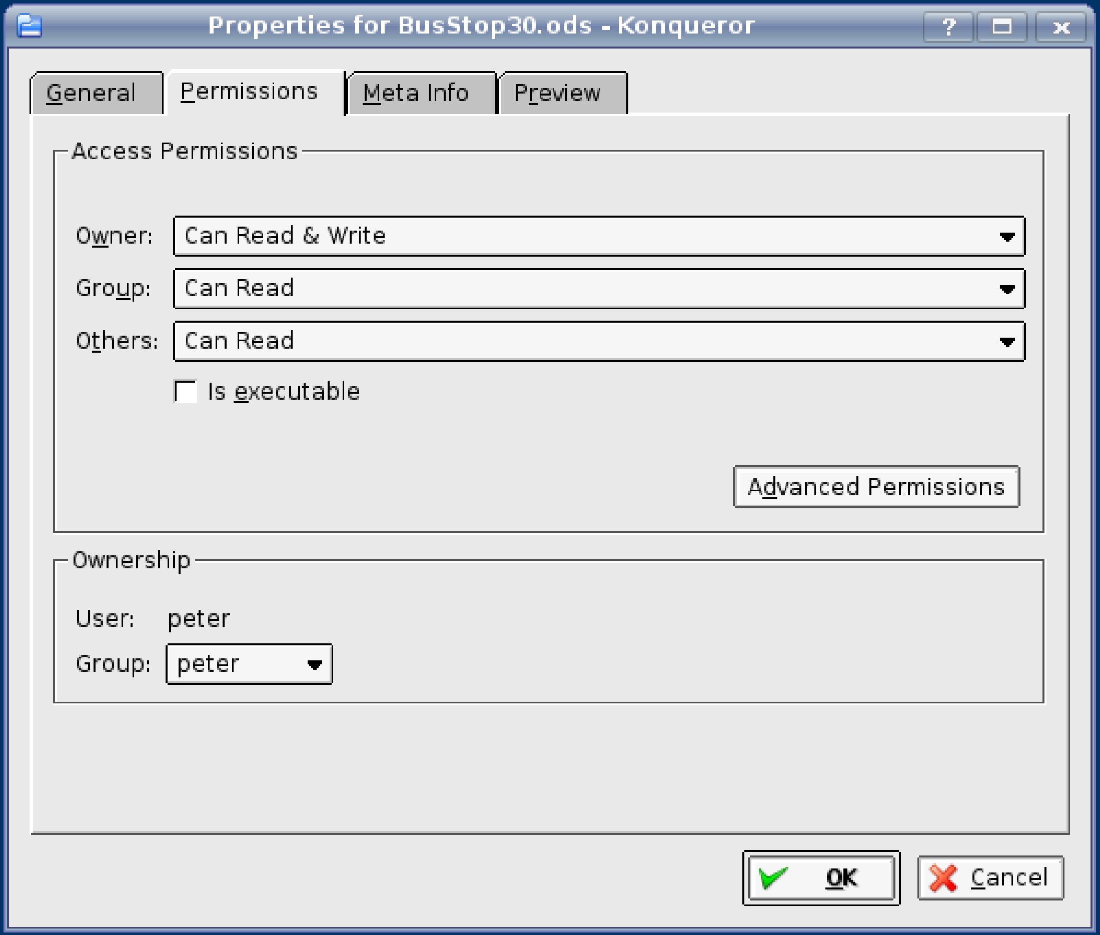
- If a directory has “set ID bits” set in its permissions, then files created in it inherit some of their properties from the directory.
- Setting the user bit (chmod u+s) means that files created in the directory will inherit the owner of the directory.
- Setting the group bit (chmod g+s) means that files created in the directory inherit the group of the directory.

## File Sticky Bit

- There is also a “file sticky bit” but according to “man chmod” it has no effect on most Linux systems.

## *Or Go All GUI*

- Find the file (browser)
- Right Click
- Select "Properties"
- Select "Permissions"


Properties for BusStop30.ods - Konqueror

Access Permissions

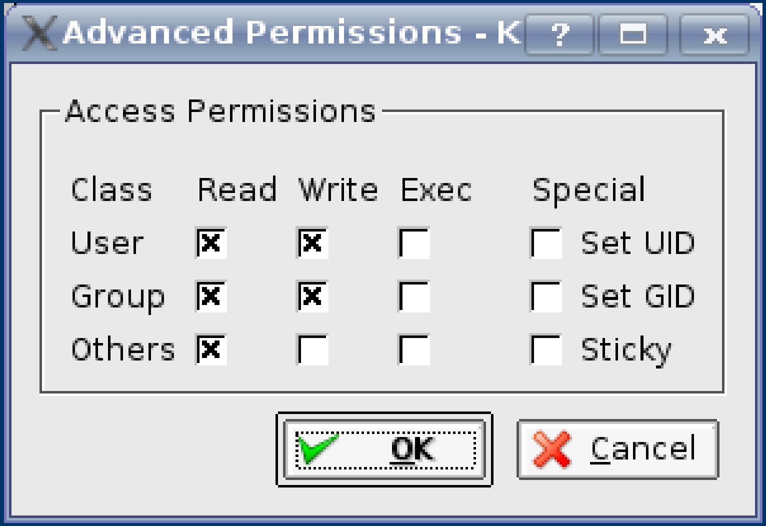
Owner: Can Read & Write
Group: Can Read
Others: Can Read
☐ Is executable

Ownership

User: peter
Group: peter

# *Advanced Permissions*

- Select advanced permissions to play with the bits directly.



Access Permissions

| Class | Read | Write | Exec | Special |
|---|---|---|---|---|
| User | ☒ | ☒ | ☐ | ☐ Set UID |
| Group | ☒ | ☒ | ☐ | ☐ Set GID |
| Others | ☒ | ☐ | ☐ | ☐ Sticky |

*All too easy, isn't it?*