

Unix and Network Programming
With Linux and C++

Nathan Warner



**Northern Illinois
University**

Computer Science
Northern Illinois University
February 16, 2023
United States

Contents

1	Commands	3
2	Permissions	4
2.1	Changing Permissions	4
2.1.1	Changing Permissions: Symbolic mode	4
2.1.2	Changing permissions: Octal mode	5
2.1.3	Exercise: Changing permissions	5
2.2	Special Permissions	5
2.2.1	Set user ID (SUID)	5
2.2.2	Set group ID (SGID)	6
2.2.3	Sticky Bit	6
2.2.4	Displaying special permissions	6
2.2.5	Setting special permissions (octal)	6
2.2.6	Setting special permissions (Symbolic)	7
2.3	User mask (umask)	7
2.3.1	Examples	8
2.4	Permissions needed for file and directory actions	8
2.4.1	Files	8
2.4.2	Directories	8
2.4.3	Both files and directories	8
3	Network Utilitys	9
3.1	Login to another computer	9
3.2	ssh	9
3.2.1	Syntax	9
3.2.2	Common options	9
3.2.3	Examples	9

3.3	Copy files to another computer	10
3.3.1	Currently in use	10
3.3.2	Secure, encrypted, part of OpenSSH	10
3.4	ftp	10
3.4.1	Syntax	10
3.4.2	Anonymous ftp	10
3.4.3	Commands	10
3.5	sftp (Secure file transfer)	11
3.5.1	Syntax	11
3.6	scp	11
3.6.1	Syntax	11
3.6.2	Common options	11
3.6.3	Examples	11

Commands

- **more, less, pg**: Display contents of file one page at a time
- **head**: Display beginning portion of file (Default: 10 lines)
- **tail**: Display end portion of file
- **wc**: Count file content (-l -w -c) (lines, words, characters)
- **diff**: Compare two files line by line
- **gzip, gunzip, zcat**: compress file content (.gz files)
- **sort**: Sort file contents (-r -n -t -k -f) (reverse, numeric, field delimiter, field1[,field2], ignore case)
- **quota -v**: Disk quota
- **lpr**: Send files to printer, -P to specify printer (lpcl, lpfl, etc)
- **lpq**: Show print queue
- **lprm**: Remove job from print queue

Permissions

Unix uses discretionary access control (DAC) model

- Each directory/file has owner
- Owner has discretion over access control details

With the exception of the super user

2.1 Changing Permissions

There are four categories regarding permissions

- User
- Group
- Other
- All

To change the permissions of a file, we use the `chmod` command

```
1  chmod -options mode file/directory
```

2.1.1 Changing Permissions: Symbolic mode

Changing Permissions: Symbolic Mode

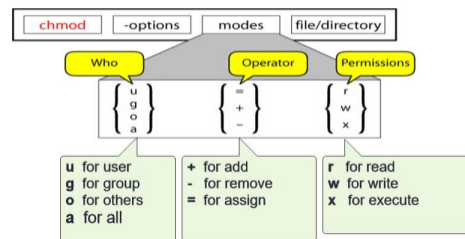


Figure 1:

Examples: Symbolic Mode

```
% chmod u-w file.txt  
% chmod u+w file.txt  
% chmod u+x script.sh  
  
% chmod g-w file.txt  
% chmod o-rw file.txt  
  
% chmod ug=rwx play.cc  
% chmod a+wx other.html  
  
% chmod u+x,go=r script.sh
```

Figure 2:

2.1.2 Changing permissions: Octal mode

Changing Permissions: Octal Mode

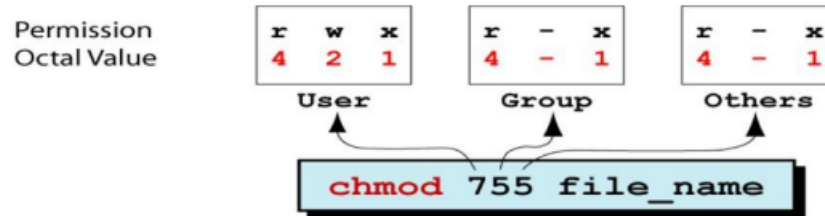


Figure 3:

2.1.3 Exercise: Changing permissions

Suppose we want to change the permissions of "myfile". We want

- Read, write, and execute for user
- Read and execute for group
- Execute for other

```
1  chmod u=rwx, g=rx, o=x myfile
2  chmod 751 myfile
```

2.2 Special Permissions

3 additional permissions can be set on files and directories

- Set user ID (SUID)
- Set group ID (SGID)
- Sticky bit

2.2.1 Set user ID (SUID)

Concept 1: SUID is used for executable files, it makes executables run with permissions of file owner, rather than invoker

For example, the passwd command uses this permission. This allows user access to otherwise protected system files while changing password

2.2.2 Set group ID (SGID)

Concept 2:

- **For executables:** The logic for SGID is the same as SUID, but for group owner rather than file owner
- **For directories:** A file created in the directory will be owned by the group owner of the directory, not the group of the user who created the file

2.2.3 Sticky Bit

Concept 3:

- **For executables:** Executable is kept in memory even after it ended
- **For directories:** Files can only be deleted by the user that created it

2.2.4 Displaying special permissions

Concept 4: The `ls -l` command does not display special permission bits. However, since special permissions require execute, they mask the execute permission when displayed with `ls -l`

2.2.5 Setting special permissions (octal)

Setting Special Permissions

suid	sgid	stb	r	w	x	r	w	x	r	w	x
4	2	1	4	2	1	4	2	1	4	2	1
7			7			7			7		
Special			user			group			others		

Use the “chmod” command with octal mode:

• `chmod 7777 filename`

Figure 4:

2.2.6 Setting special permissions (Symbolic)

Setting Special Permissions

- chmod with symbolic notation:

u+s	add SUID
u-s	remove SUID

g+s	add SGID
g-s	remove SGID

+s	add SUID and SGID
+t	set sticky bit

Figure 5:

2.3 User mask (umask)

File mode creation mask

- umask (user mask)
 - governs default permission for files and directories
 - sequence of 9 bits: 3 times 3 bits of rwx
 - default:

000	000	010	(002)
000	010	010	(022)

 on turing/hopper
- in octal form its bits are removed from:
 - for a file:

110	110	110	(666)
-----	-----	-----	-------
 - for a directory:

111	111	111	(777)
-----	-----	-----	-------
- permission for new
 - file:

110	110	100	(664)
-----	-----	-----	-------
 - directory:

111	111	101	(775)
-----	-----	-----	-------

Figure 6:

2.3.1 Examples

User Mask value examples

	Directory Default: 777	File Default: 666
000	777 (rwx rwx rwx)	666 (rw- rw- rw-)
111	666 (rw- rw- rw-)	666 (rw- rw- rw-)
222	555 (r-x r-x r-x)	444 (r-- r-- r--)
022	755 (rwx r-x r-x)	644 (rw- r-- r--)
002	775 (rwx rwx r-x)	664 (rw- rw- r--)
066	711 (rwx --x --x)	600 (rw- --- ---)
666	111 (--x --x --x)	000 (--- --- ---)
777	000 (--- --- ---)	000 (--- --- ---)

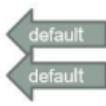


Figure 7:

2.4 Permissions needed for file and directory actions

2.4.1 Files

- **Read:** View file contents (open, read)
- **Write:** Change file contents
- **Execute:** Run executable file

2.4.2 Directories

- **Read:** List directory contents (Only names)
- **Write:** Change directory contents (need execute aswell)
- **Execute:** Make it current directory, search for files in it
- **Renaming files:** Write and execute permissions on the directory
- **Deleting files:** Write and execute permissions on the directory

2.4.3 Both files and directories

- **Moving files:** Read permissions on file and write permissions on target directory

Network Utilitys

- Login to another computer
 - telnet, rlogin, rsh, ssh
- Copy files to another computer
 - scp
 - ftp, sftp

3.1 Login to another computer

- telnet rlogin, rsh no longer used
 - Transmit username/password without encryption
- ssh
 - Invokes shell on remote computer securely
 - **Used to:** Remote login and run command on remote computer

3.2 ssh

3.2.1 Syntax


```
1  ssh [user@]hostname [command]
```

This command logs in user to hostname, or if command is given, runs it on remote host

3.2.2 Common options

- **-l:** login-name
- **-X:** enable X11 forwarding

3.2.3 Examples



```
% ssh turing.cs.niu.edu
% ssh z123456@hopper.cs.niu.edu
% ssh z123456@hopper.cs.niu.edu w
% ssh -X turing.cs.niu.edu -l z123456
% ssh -X ege@turing.cs.niu.edu thunar
```

Figure 8:

3.3 Copy files to another computer

3.3.1 Currently in use

- ftp

3.3.2 Secure, encrypted, part of OpenSSH

- **sftp**: Secure file transfer
- **scp**: Secure copy to remote host

3.4 ftp

3.4.1 Syntax

```
1 ftp hostname
```

This will prompt for userid and password

3.4.2 Anonymous ftp

- **Userid**: ftp or anonymous
- **Password**: Your email address

3.4.3 Commands

- **help**
- **ls**
- **cd**
- **put, get**
 - copy a file from local to remote host, or vice versa
- **mput, mget**
 - put/get multiple files, can use wildcards
- **bye**

3.5 sftp (Secure file transfer)

3.5.1 Syntax

```
1 sftp user@hostname
```

- Will prompt for password
- Same commands as ftp

3.6 scp

3.6.1 Syntax

```
1 scp source target
```

- source and target use extended form of pathname

```
1 user@host:pathname
```

3.6.2 Common options

- **-r**: Recursively copy entire directories
- **-C**: Enables compression
- **-l**: Limit bandwidth, specified in Kbit/s

3.6.3 Examples

```
1 scp screenshot.png z123456@turing.cs.niu.edu:  
2 scp z123456@hopper.cs.niu.edu:assign1.cc .
```