**Basic bash Shell Commands**

/etc/passwd file

These fields are:

■ The username

■ The user’s password (or a placeholder if the password is stored in another file

■ The user’s system user ID number

■ The user’s system group ID number

■ The user’s full name

■ The user’s default home directory

■ The user’s default shell program

**Information from prompt:**

[girish@mypc ~]$

■ The username that started the shell

■ The current virtual console number

■ The current directory (the tilde sign is shorthand for the home directory)

■ PS1: Controls the format of the default command line prompt

■ PS2: Controls the format of the second-tier command line prompt

girish@1[~]$ echo $PS1

\u@\l[\W]\$

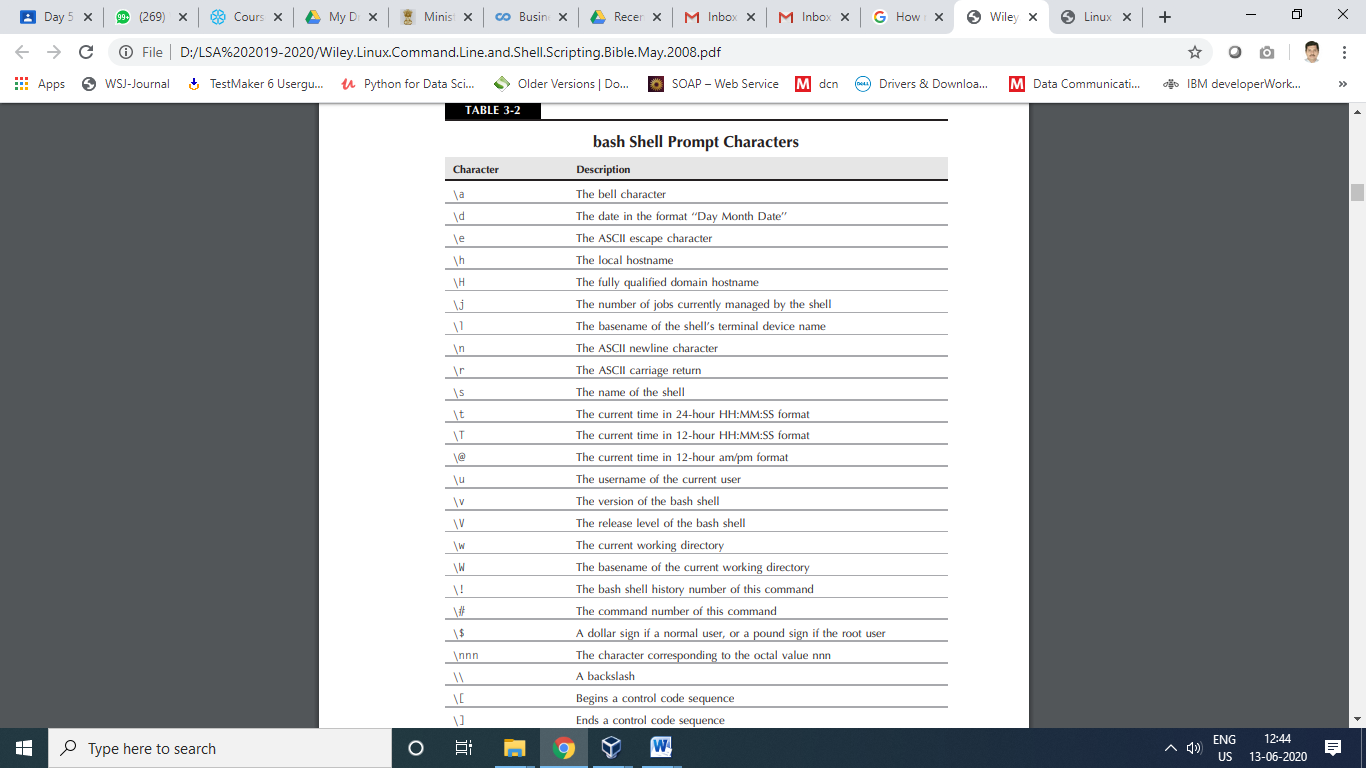
girish@1[~]$ echo $PS2

>

girish@1[~]$

DEFAULT=$PS1

$ PS1="[custom text] \u@\h:\w\$ "



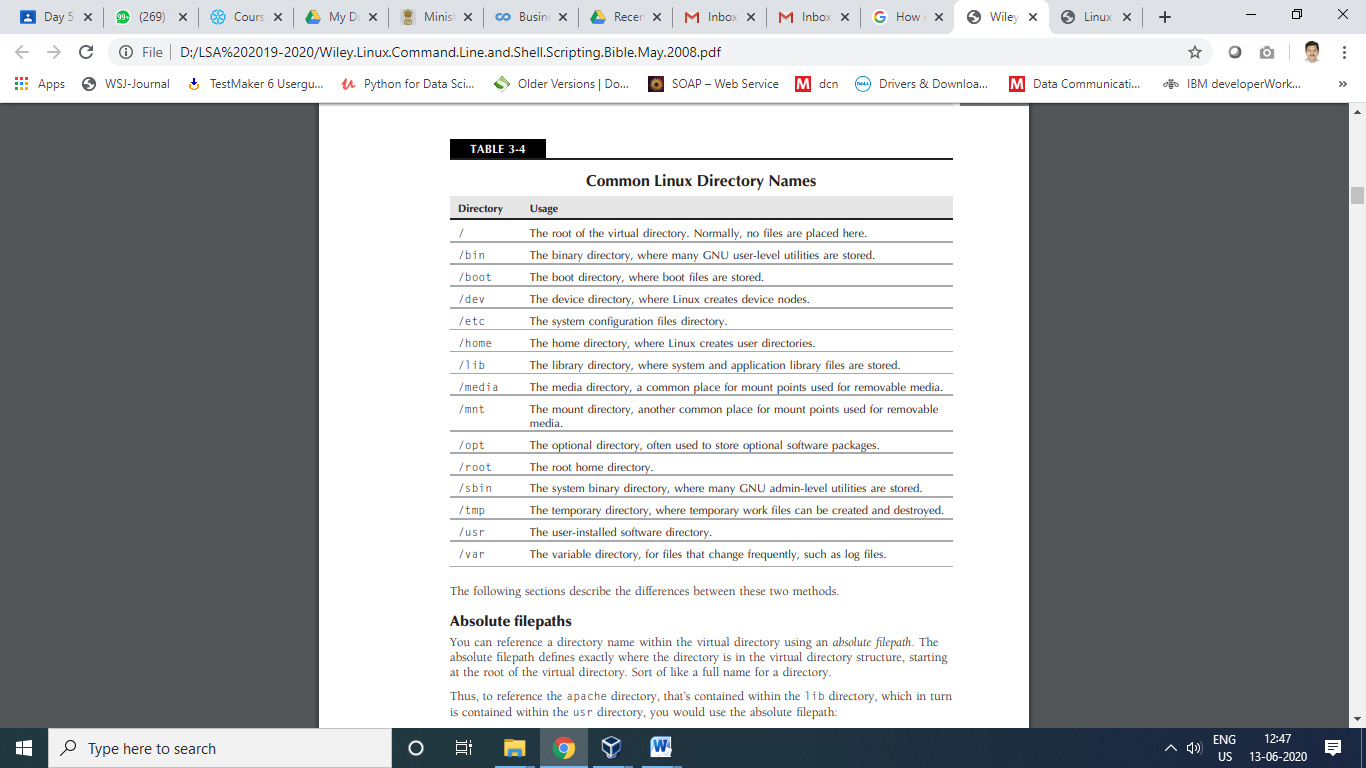
Linux file system

cd destination

The destination parameter, though, can be expressed using two different methods:

■ An absolute filepath

■ A relative filepath



**Relative filepaths**

■ The dot (.) to represent the current directory

■ The double dot (..) to represent the parent directory

**File and Directory Listing**

**ls**

ls -F

The -F parameter flags the directories with a forward slash, to help identify them in the

listing. Similarly, it flags executable files (like the myprog file above) with an asterisk, to help you

find the files that can be run on the system easier.

ls –R ???

ls –l

ls –a

The **long listing** format lists each file and directory contained in the directory on a single line.

Besides the filename, it shows additional useful information. The first line in the output shows

the total number of blocks contained within the directory. Following that, each line contains the following information about each file (or directory):

■ The file type (such as directory (d), file (-), character device (c), or block device (b)

■ The permissions for the file

■ The number of hard links to the file

■ The username of the owner of the file

■ The group name of the group the file belongs to

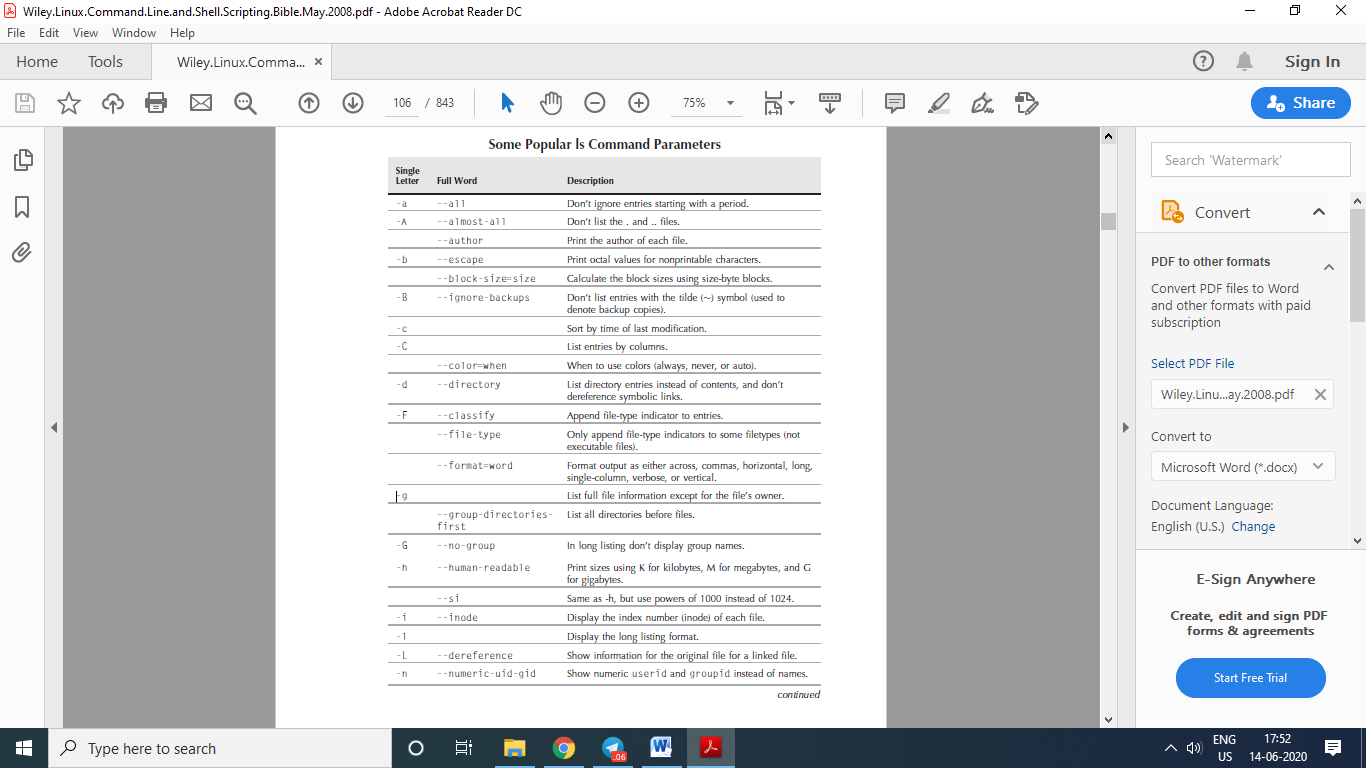
■ The size of the file in bytes

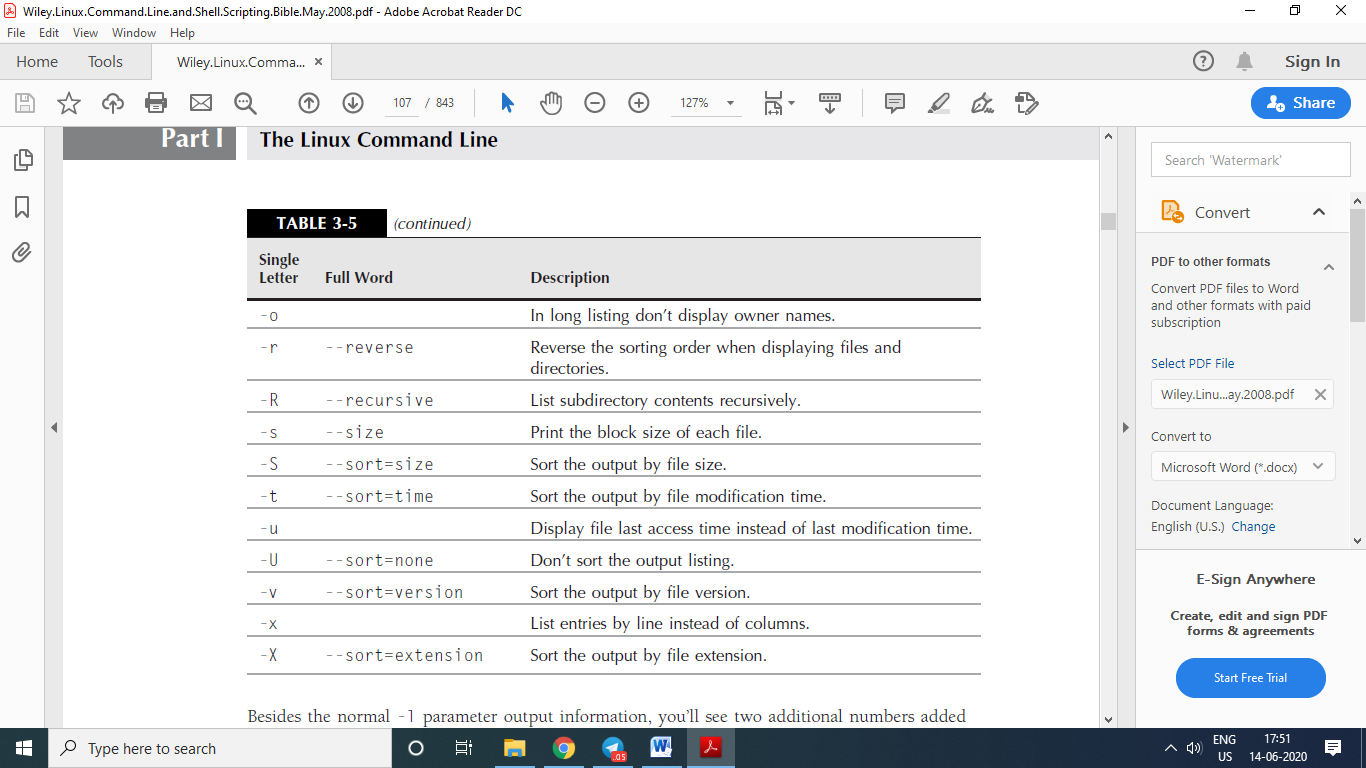
■ The time the file was modified last

■ The file or directory name

Explain following command:

ls –sail





**Filtering listing output**

$ ls -l myprog

The ls command also recognizes standard wildcard characters and uses them to match patterns within the filter:

■ A question mark to represent one character

■ An asterisk to represent zero or more characters

$ ls -l mypro?

$ ls -l myprob\*

**File Handling**

**Creating files**

$ touch test1

Specify the specific time stamp 2020, June, 20 08.20 am for creating a file greet.

**Copying files**

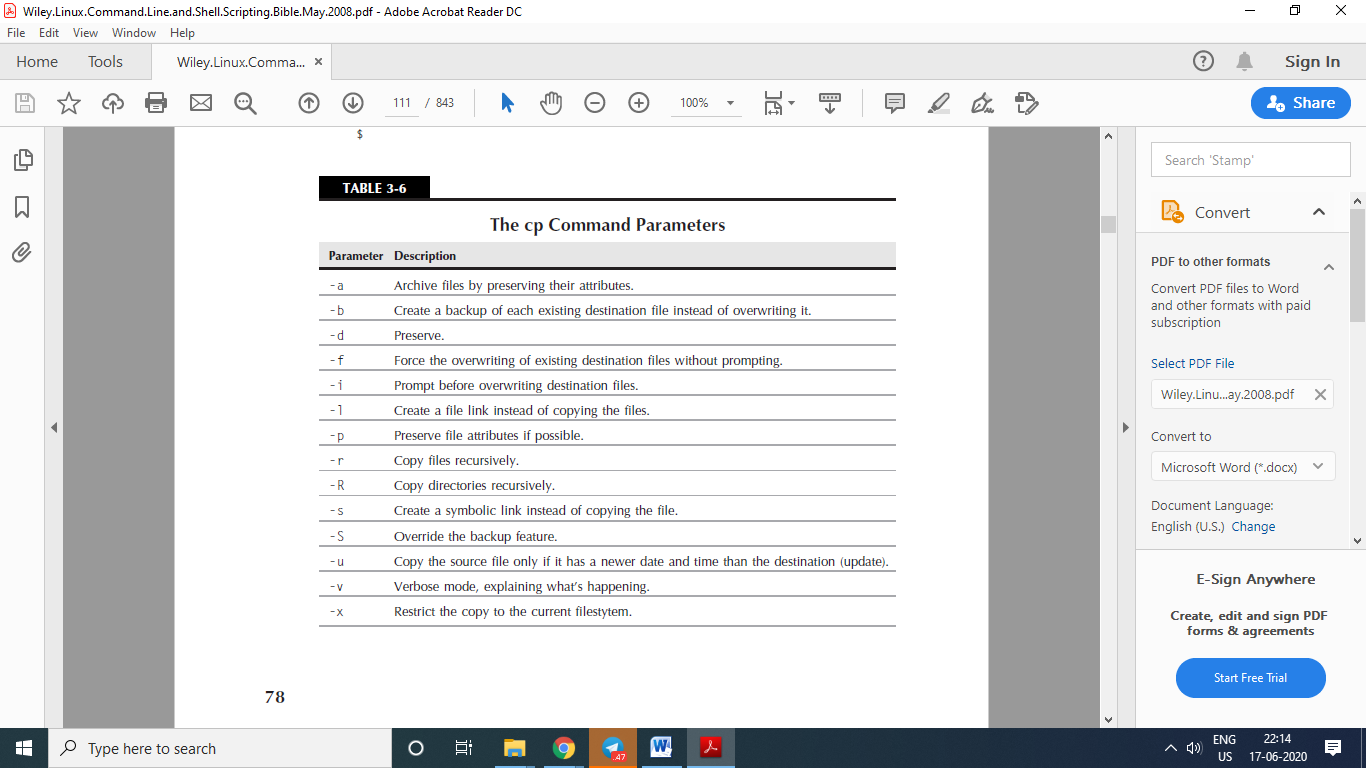
cp source destination

$ cp test1 test2

The -R parameter is extremely powerful. It allows you to recursively copy the contents of an

entire directory in one command:

$ cp -R dir1 dir2



**Linking files**

■ A symbolic, or soft, link

■ A hard link

$ cp -l test1 test4

**You can only create a hard link between files on the same physical medium. You can’t**

**create a hard link between files under separate mount points. In that case, you’ll have**

**to use a soft link.**

On the other hand, the -s parameter creates a symbolic, or soft, link:

$ cp -s test1 test5

**Instead of using the cp command, if you want to link files you can also use the ln**

**command. By default the ln command creates hard links. If you want to create a soft**

**link, you’ll still need to use the -s parameter.**

**Renaming files**

$ mv test2 test6

What is dangling pointer?

$ mv dir2 dir4

**Deleting files**

rm

rm –f

rm –r

rm file1.txt file2.txt file3.txt file4.txt

rm -i try1.txt

use ‘**-d**‘ option in rm command to delete an empty directory.

rm -R

**Directory Handling**

**Creating directories**

$ mkdir dir3

**Deleting directories**

$ rmdir dir3

$ rm –r xyz

$ rm –rf abc

$ head

$ tail

**Understanding Linux File Permissions**

**Linux Security**

**username / passwd**

**The /etc/passwd**

■ The login username

■ The password for the user

■ The numerical UID of the user account

■ The numerical group ID (GID) of the user account

■ A text description of the user account (called the comment field)

■ The location of the HOME directory for the user

■ The default shell for the user

**The /etc/shadow file**

■ The login name corresponding to the login name in the /etc/passwd file

■ The encrypted password

■ The number of days since January 1, 1970 that the password was last changed

■ The minimum number of days before the password can be changed

■ The number of days before the password must be changed

■ The number of days before password expiration that the user is warned to change the password

■ The number of days after a password expires before the account will be disabled

■ The date (stored as the number of days since January 1, 1970) since the user account was disabled

■ A field reserved for future use

**Adding a new user**

The useradd command uses a combination of system default values and command line parameters to define a user account. To see the system default values used on your Linux

distribution, enter the useradd command with the -D parameter:

# /usr/sbin/useradd -D

GROUP=100

HOME=/home

INACTIVE=-1

EXPIRE=

SHELL=/bin/bash

SKEL=/etc/skel

CREATE\_MAIL\_SPOOL=yes

This example shows the following default values:

■ The new user will be added to a common group with group ID 100.

■ The new user will have a HOME account created in the directory /home/loginname.

■ The account will not be disabled when the password expires.

■ The new account will not be set to expire at a set date.

■ The new account will use the bash shell as the default shell.

■ The system will copy the contents of the /etc/skel directory to the user’s HOME directory.

■ The system will create a file in the mail directory for the user account to receive mail.

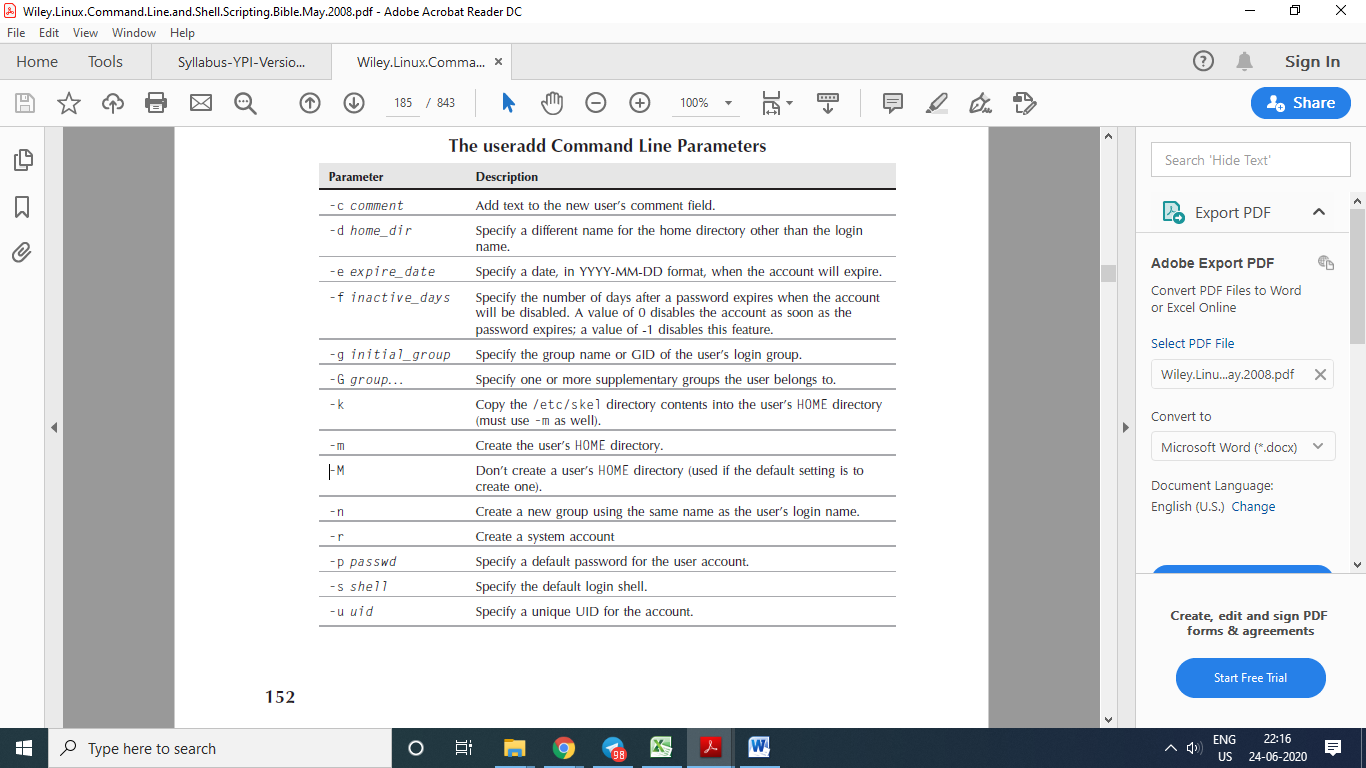
# ls -al /etc/skel

…

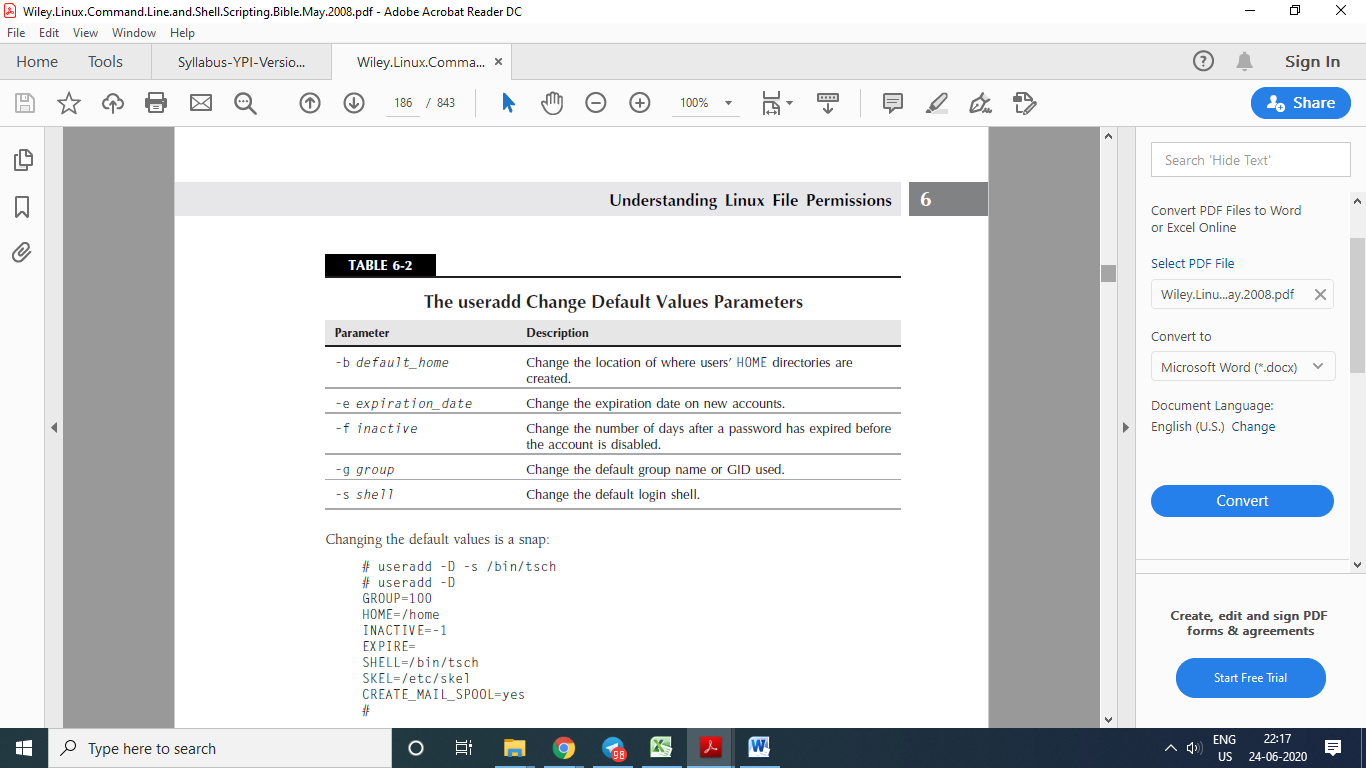
These are the standard startup files for the bash shell environment. The system automatically copies these default files into every user’s HOME directory you create.

# /usr/sbin/useradd test

# ls -al /home/test



You can change the system default new user values by using the -D parameter



# useradd -D -s /bin/tsch

# useradd -D

GROUP=100

HOME=/home

INACTIVE=-1

EXPIRE=

SHELL=/bin/tsch

SKEL=/etc/skel

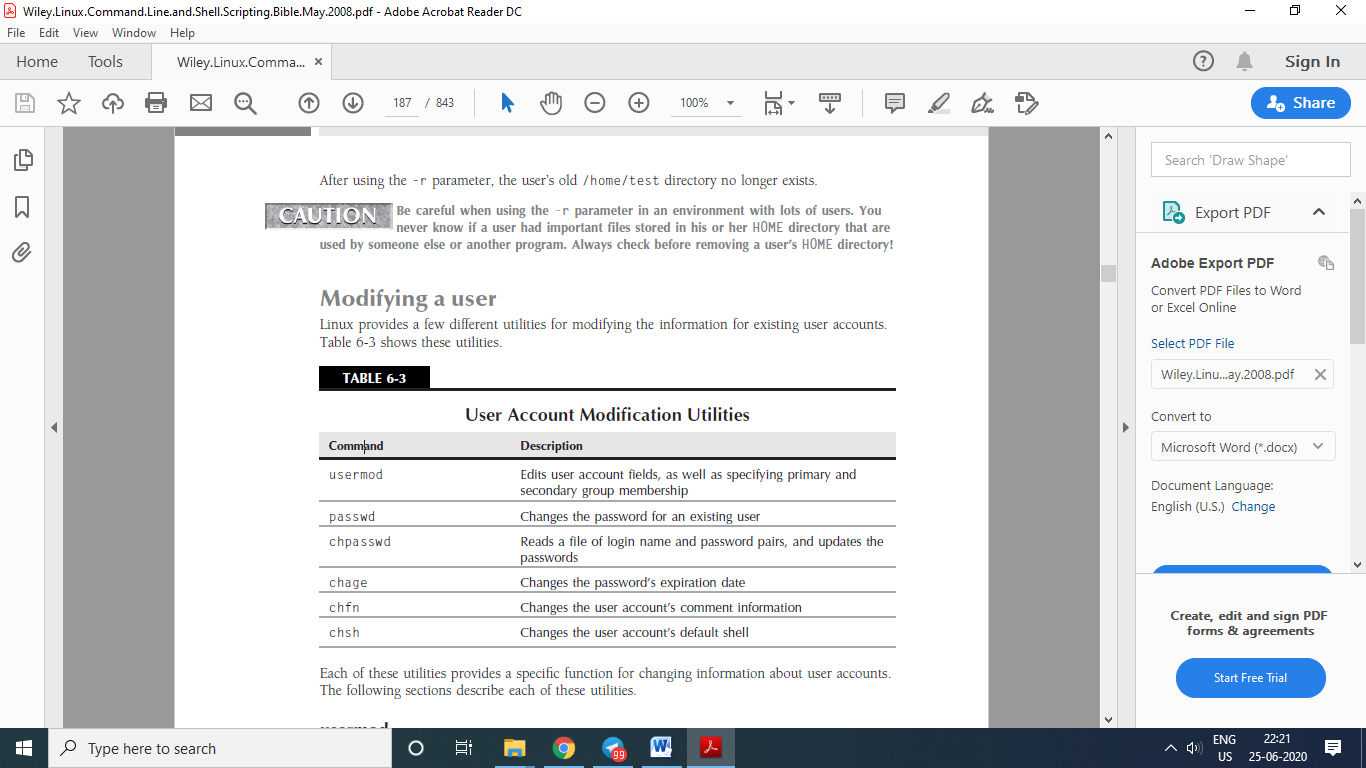
CREATE\_MAIL\_SPOOL=yes

**Removing a user**

# /usr/sbin/userdel -r test

-r parameter: userdel will remove the user’s HOME directory, along with the user’s mail directory.

**Modifying a user**



**usermod**

The usermod command is the most robust of the user account modification utilities. It provides options for changing most of the fields in the /etc/passwd file.

■ -l to change the login name of the user account

■ -L to lock the account so the user can’t log in

■ -p to change the password for the account

■ -U to unlock the account so that the user can log in

**passwd and chpasswd**

# passwd test

Changing password for user test.

New UNIX password:

Retype new UNIX password:

passwd: all authentication tokens updated successfully.

#

The -e option is a handy way to force a user to change the password on the next log in.

The chpasswd command reads a list of login name and password pairs (separated by a colon) from the standard input, and automatically encrypts the password and sets it for the user account.

**chsh, chfn, and chage**

The chsh, chfn, and chage utilities are specialized for specific functions. The chsh command

allows you to quickly change the default login shell for a user. You must use the full pathname for the shell, and not just the shell name:

# chsh -s /bin/csh test

Changing shell for test.

Shell changed.

The chfn command provides a standard method for storing information in the comments field in

the /etc/passwd file.

The finger command allows you to easily find information about people on your Linux system:

# finger girish

If you use the chfn command with no parameters, it queries you for the appropriate values to

enter in to the comment field:

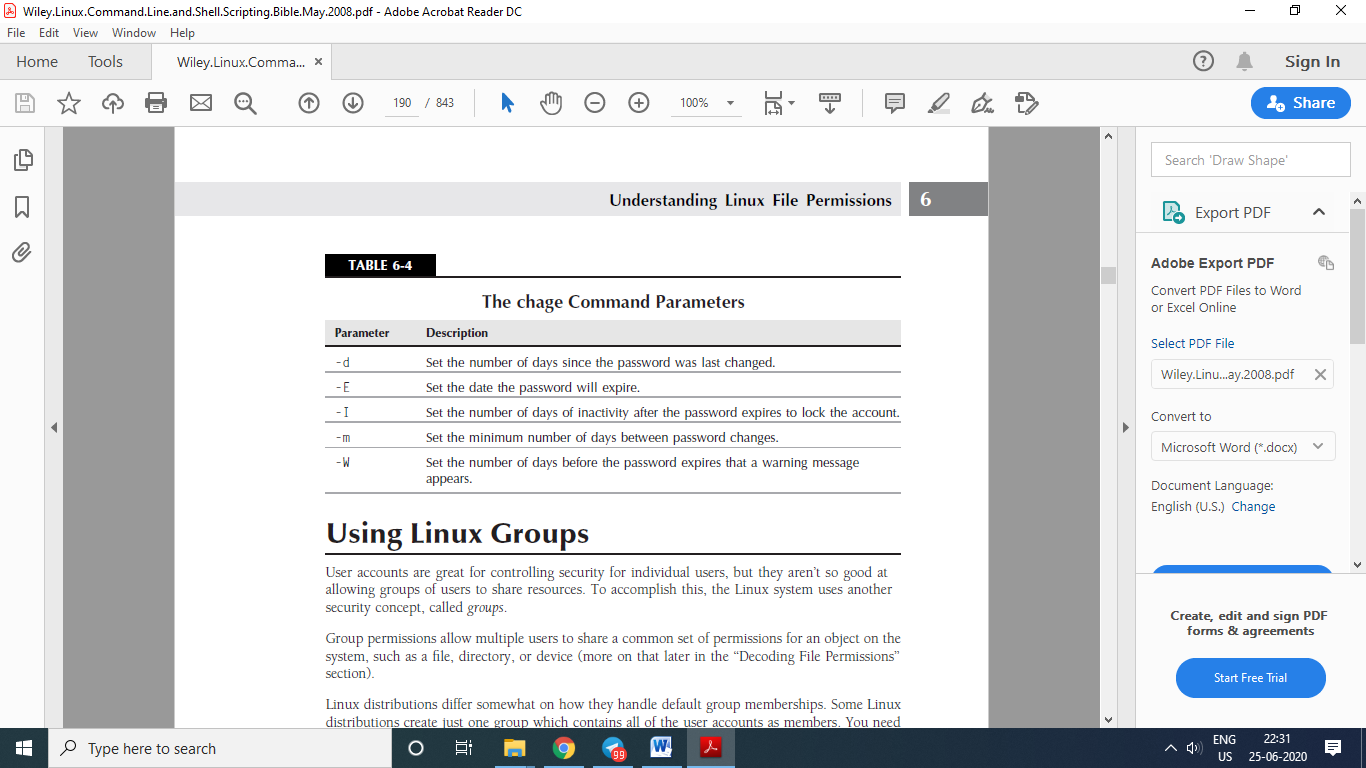
# chfn test

Finally, the chage command helps us manage the password aging process for user accounts.

The chage date values can be expressed using one of two methods:

■ A date in YYYY-MM-DD format

■ A numerical value representing the number of days since January 1, 1970



**Group**

/etc/group

■ The group name

■ The group password

■ The GID

■ The list of user accounts that belong to the group

groupadd

usermod –G shared girish

groupmod –g

groupmod –n

groupdel

**File permissions**

ls –l

first character can be

-

d

l

c

b

n

there are three sets of three characters. Each set of three characters defines an access permission triplet:

■ r for read permission for the object

■ w for write permission for the object

■ x for execute permission for the object

If a permission is denied, a dash appears in the location. The three sets relate the three levels of security for the object:

■ The owner of the object

■ The group that owns the object

■ Everyone else on the system

**Default file permissions**

$ touch newfile

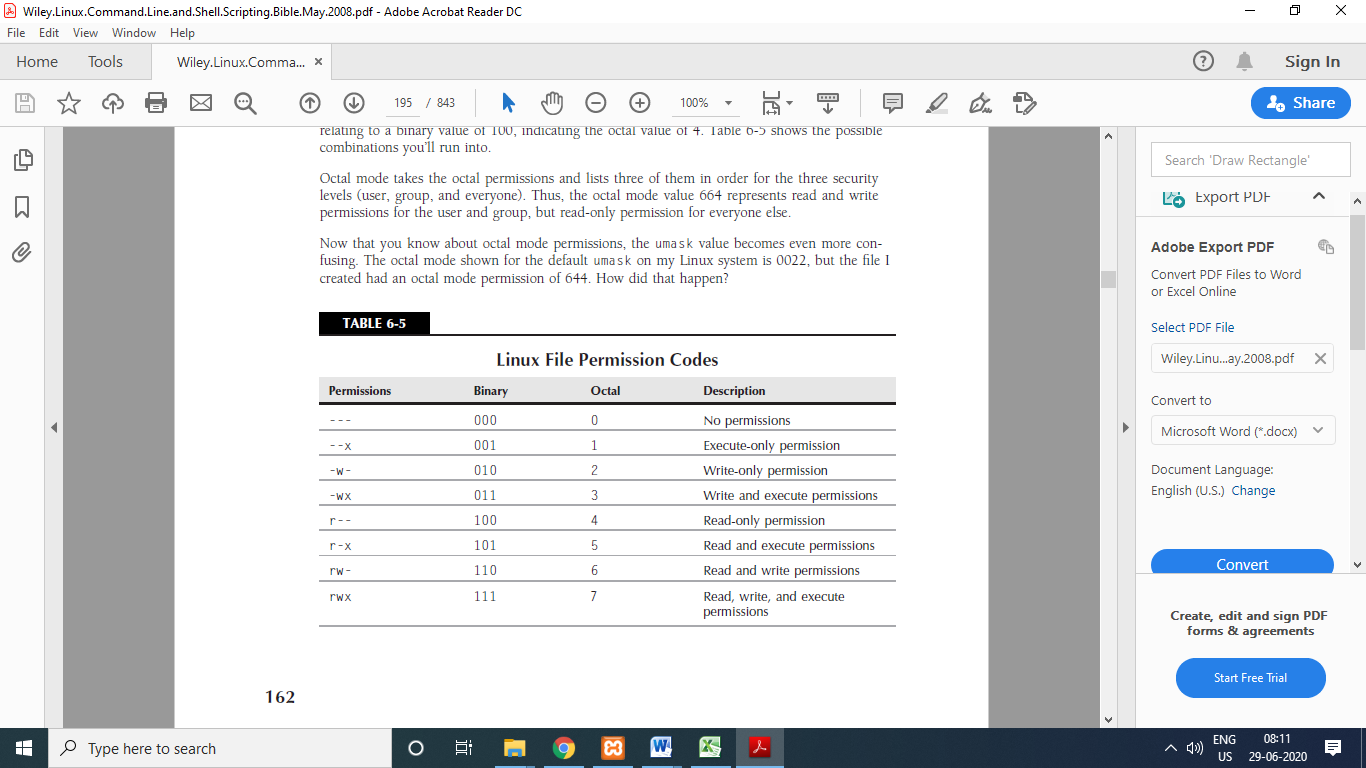
$ ls -al newfile

The touch command created the file using the default permissions assigned to my user

account. The umask command shows and sets the default permissions:

$ umask

0022



$ umask 026

$ touch newfile2

**Changing Security Settings**

**Changing permissions**

$chmod *options mode file*

$ chmod 760 newfile

The format for specifying a permission in symbolic mode is:

[ugoa...][[+-=][rwxXstugo...]

The first group of characters defines to whom the new

permissions apply:

■ u for the user

■ g for the group

■ o for others (everyone else)

■ a for all of the above

$ chmod o+r newfile

$ chmod u-x newfile

**Changing ownership**

Why ownership of file need to be changed?

chown *options owner[.group] file*

# chown tycs newfile

# chown tycs.tycs newfile

# chown .girish newfile

# chown test. newfile ???

Use of -R , -h, wildcard characters like \*, ?

**Only the root user can change the owner of a file. Any user can change the default group of a file, but the user must be a member of the groups the file is changed from and to.**

The chgrp command provides an easy way to change just the default group for a file or directory:

$ chgrp shared newfile

$ ls -l newfile

-rw-rw-r-- 1 rich shared 0 Sep 20 19:16 newfile

$

Now any member in the shared group can write to the file. This is one way to share files on a

Linux system.

**Sharing Files**

When you create a new file, Linux assigns the file permissions of the new file using your default UID and GID. To allow others access to the file, you need to either change the security permissions for everyone security group or assign the file a different default group that contains other users.

There are three additional bits of information that Linux stores for each file and directory:

■ The set user id (SUID): When a file is executed by a user, the program runs under the permissions of the file owner.

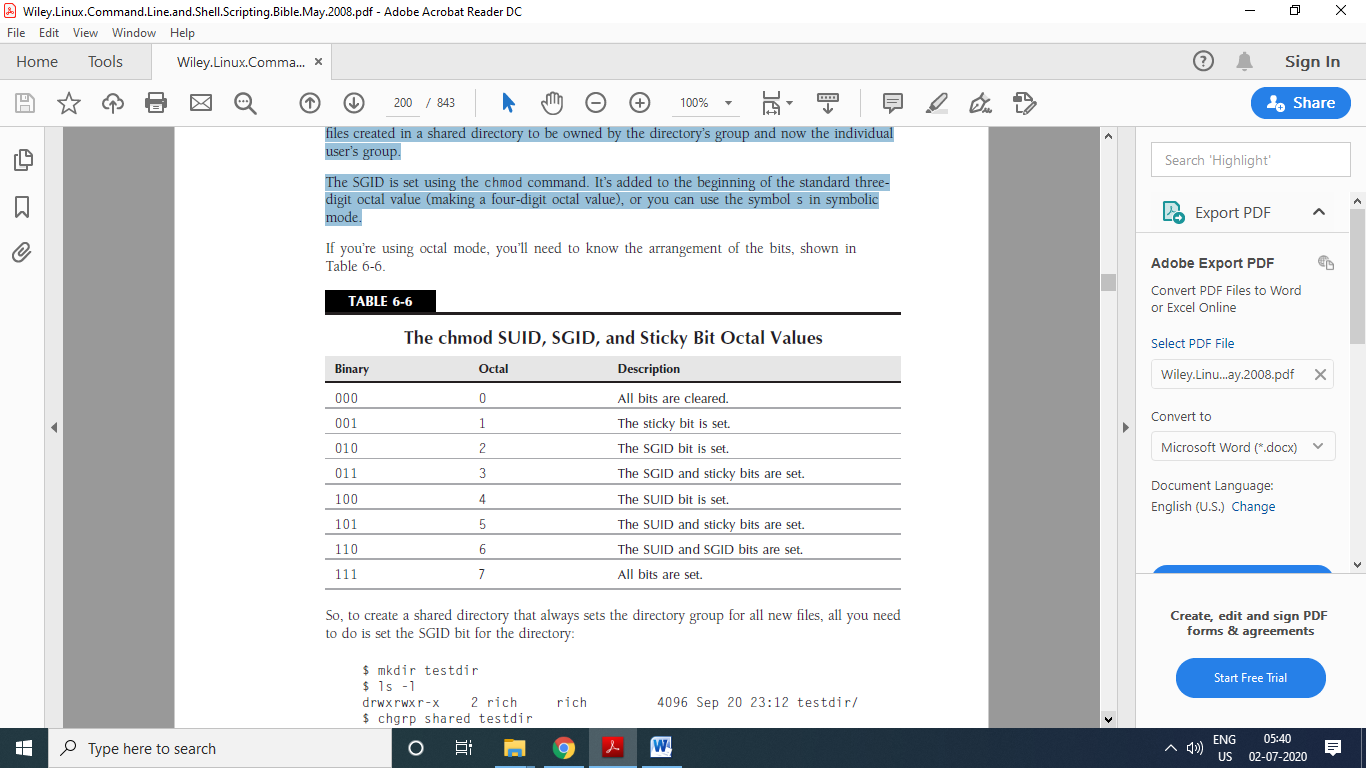
■ The set group id (SGID): For a file, the program runs under the permissions of the file

group. For a directory, new files created in the directory use the directory group as the default group.

■ The sticky bit: The file remains (sticks) in memory after the process ends.

The SGID bit is important for sharing files. By enabling the SGID bit, you can force all new files created in a shared directory to be owned by the directory’s group and not the individual user’s group.

The SGID is set using the chmod command. It’s added to the beginning of the standard three digit octal value (making a four-digit octal value), or you can use the symbol s in symbolic mode.



What is SUID and SGID?

SUID is a special file permission for executable files which enables other users to run the file with effective permissions of the file owner. Instead of the normal **x** which represents execute permissions, you will see an **s** (to indicate SUID) special permission for the user.

SGID is a special file permission that also applies to executable files and enables other users to inherit the effective GID of file group owner. Likewise, rather than the usual **x** which represents execute permissions, you will see an **s** (to indicate SGID) special permission for group user.

So, to create a shared directory that always sets the directory group for all new files, all you need to do is set the SGID bit for the directory:

$ mkdir testdir

$ ls -l

drwxrwxr-x 2 girish girish 4096 Sep 20 23:12 testdir/

$ chgrp shared testdir

$ chmod g+s testdir

$ ls -l

drwxrwsr-x 2 girish shared 4096 Sep 20 23:12 testdir/

$ umask 002

$ cd testdir

$ touch testfile

$ ls -l

total 0

-rw-rw-r-- 1 girish shared 0 Sep 20 23:13 testfile

$

The first step is to create a directory that you want to share using the mkdir command. Next, the chgrp command is used to change the default group for the directory to a group that contains the members who need to share files.

Finally, the SGID bit is set for the directory, to ensure that any files created in the directory use the shared group name as the default group.

For this environment to work properly, all of the group members need to have their umask values set to make files writable by group members. This is why I changed my umask to 002.