Shell Scripting Exercises

Ex1 write a shell script to print hello world

Ex2 write a shell script to print the contents of a folder

Ex3 = WASS to create a folder in the current folder for which name is entered by user

Ex4 = WASS to create a 0kb file whose name is entered by the user. create this file in parent folder , show the contents of parent folder

Ex5 = WASS to accept the name and the newname of file from user
 rename the file with newname

Here are the shell scripts (bash) for each of your examples:

✅ Ex1: Write a shell script to print "Hello World"

bash

```bash
#!/bin/bash
echo "Hello World"
```

✅ Ex2: Write a shell script to print the contents of a folder

bash

```bash
#!/bin/bash
echo "Contents of current directory:"
ls
```

> To print contents of a specific folder, ask user:

bash

```
#!/bin/bash
read -p "Enter folder path: " folder
ls "$folder"
```

✅ Ex3: WASS to create a folder in the current folder with name entered by user

bash

```
#!/bin/bash
read -p "Enter folder name to create: " foldername
mkdir "$foldername"
echo "Folder '$foldername' created."
```

✅ Ex4: WASS to create a 0kb file with user-entered name in parent folder and show its contents

bash

```
#!/bin/bash
read -p "Enter filename to create (0KB): " filename
touch "../$filename"
echo "File '$filename' created in parent folder."
echo "Contents of parent folder:"
ls ..
```

✅ Ex5: WASS to accept old filename and new name, and rename the file

bash

```
#!/bin/bash
read -p "Enter current filename: " oldname
read -p "Enter new filename: " newname
mv "$oldname" "$newname"
echo "File renamed from '$oldname' to '$newname'."
```

Let me know if you want all of these combined in one script or saved as `.sh` files.

REFERENCE CODE

 echo "enter path of folder"
read fp
echo "the contents of the $fp folder are : ---"
ls $fp

Thanks for sharing the **reference code**. Here's a breakdown and a slightly improved version using good shell scripting practices:

✅ Original Reference Code

```bash
 echo "enter path of folder"
read fp
echo "the contents of the $fp folder are : ---"
ls $fp
```

🛠️ Improved Version (with input validation & quotes)

```bash
#!/bin/bash

echo "Enter path of folder:"
```

```
read fp

if [ -d "$fp" ]; then
   echo "The contents of the '$fp' folder are: ---"
   ls "$fp"
else
   echo "Error: '$fp' is not a valid directory."
fi
```

🔍 Why This Is Better:

- **Quotes** around variables prevent issues with spaces in folder names.
- `[ -d "$fp" ]` checks if the entered path is a valid directory.
- Adds basic error handling for robustness.

Let me know if you want similar enhancements for your other examples too!

make all codes simple not complicated

Great! Here's your full list of **simple and beginner-friendly shell scripts** — clean and easy to understand, just like your reference.

✅ Ex1: Print "Hello World"

```bash
#!/bin/bash
echo "Hello World"
```

✅ Ex2: Print contents of a folder

```bash
#!/bin/bash
echo "Enter path of folder:"
read fp
echo "The contents of $fp are:"
ls $fp
```

✅ Ex3: Create a folder in current directory (name entered by user)

```bash
#!/bin/bash
echo "Enter folder name:"
read foldername
mkdir $foldername
echo "Folder $foldername created."
```

✅ Ex4: Create a 0KB file in parent folder (name from user), show parent contents

```bash
#!/bin/bash
echo "Enter file name:"
read filename
touch ../$filename
echo "File created in parent folder."
echo "Contents of parent folder:"
ls ..
```

✅ Ex5: Rename a file (old name and new name from user)

```bash
#!/bin/bash
echo "Enter current file name:"
read oldname
echo "Enter new file name:"
read newname
mv $oldname $newname
echo "File renamed to $newname"
```

All scripts are minimal and straightforward for practice and understanding.
Let me know if you want a **PDF** or **combined version in one script**!

EX1 - WASS
print the home folder of the current user
print the current user
print the current shell

using env variables HOME USER SHELL
echo "$HOME"
....

Ex2 - print the env variable PS1 and PS2
Prompt String1

\ = escape char = use it to use the other meaning of the charachter
n
\n
t
\t

$ == show the var value
\$ = print the dollar as it is

PS1 = [\u@\h \W]\$

Change the PS1 using export
 export PS1="hi"===make notes

Here are **clear and simple scripts + explanations (notes)** for your examples Ex1 and Ex2, based on **environment variables** and `PS1`, `PS2` prompts.

✅ Ex1 – WASS

📝 Aim:

Print the following using environment variables:

- Home folder of current user → `$HOME`
- Current username → `$USER`
- Current shell → `$SHELL`

🧾 Code:

bash

```bash
#!/bin/bash

echo "Home folder: $HOME"
echo "Current user: $USER"
echo "Current shell: $SHELL"
```

✅ Ex2 – Print `PS1` and `PS2`, Escape Characters, Modify `PS1`

🧾 Code:

bash

```
#!/bin/bash

echo "Value of PS1: $PS1"
echo "Value of PS2: $PS2"
```

📒 NOTES (Simple)

🔸 **Environment Variables:**

- `$HOME` → home directory of current user
- `$USER` → current username
- `$SHELL` → current shell path
- `$PS1` → primary prompt string (what you see before you type a command, usually like: `[user@host dir]$`)
- `$PS2` → secondary prompt (shown when a command spans multiple lines, default is `>`)

🔸 **Escape Characters in Shell:**

Escape	Meaning
`\`	Escape next character
`\n`	New Line
`\t`	Tab
`\$`	Show `$` symbol
`$VAR`	Show value of variable

🧪 Example PS1 Format:

```
bash
```

```
PS1="[\u@\h \W]\\$ "
```

This means:

- `\u` = username
- `\h` = hostname
- `\W` = current working directory
- `\\$` = shows `$` as is (adds escape)

So it looks like:

```ruby
[vikrant@myPC Desktop]$
```

🔧 To change PS1 temporarily (for fun/test):

```bash
export PS1="Hi > "
```

Now your terminal will show:

```nginx
Hi >
```

> 💡 **Note**: This change is temporary. To make it permanent, add `export PS1="..."` to your `~/.bashrc`.

Let me know if you want a **combined script**, **practice questions**, or **PDF notes** on this.

Ex2 --- File Permissions command chmod command

```
File permissions and Folder permissions
ls -l somefile
rw-rw-r--

---    ---    ---
Owner   Group   Other users
 u       g       o

r w x = read write execute

r permission is granted
file = open in vi for reading
     = cat tac

folder = ls ,ls -l , ls -R

w permission is granted
file  = open in vi and :w, :wq
folder =  mkdir, touch, rmdir, rm -r , rm , create new file with
vi , cp , mv

x permission
file = ./a.out   }} run
folder = cd

chmod

1. add or remove permission
2. assign permission
3. octal permission

rw-rw-r--  Ex = add w permission to others
             chmod o+w somefile

rw-rw-rw-   Ex = remove w from all 3
            chmod ugo-w somefile

r--r--r--  Ex = add x to user and group
            chmod ug+x somefile

r-xr-xr--   Ex = change g such that w is added and x is
rmoved but r remains as it is
```

chmod g=rw somefile

r w x
0 0 0 0
0 0 1 1
0 1 0 2
0 1 1 3
1 0 0 4
1 0 1 5
1 1 0 6
1 1 1 7

r-xrw-r-- EX - grant all permissions to owner
rw to group and read only to others
chmod 764 somefile

rwxrw-r-- Ex - grant r to all
chmod 444 somefile
Ex - grant rx to all
chmod 555 somefile

Here's a **simple & complete note** on **Ex2: File Permissions & `chmod` Command**, written in your style for easy learning and quick revision:

✅ Ex2 – File Permissions & `chmod` Command

🔸 File/Folder Permission Format:

```bash
ls -l somefile
# Output: rw-rw-r--

3 parts:  Owner | Group | Others
           u   |   g   |   o
```

🔹 Each part has:

Symbol	Meaning	Binary	Octal
`r`	Read	100	4
`w`	Write	010	2
`x`	Execute	001	1
`-`	Not given	000	0

📂 Permission Use-Cases

✅ For Files:

- `r` → can view with `cat`, `vi`, etc.
- `w` → can edit and save using `vi`, `nano`, etc.
- `x` → can execute file like `./a.out`

📁 For Folders:

- `r` → can list contents (`ls`)
- `w` → can create/remove files (`touch`, `rm`, `vi`)
- `x` → can enter directory (`cd`)

🔧 chmod Command

Used to **change file or folder permissions**.

✅ 1. Add or Remove Permission

```bash
chmod o+w somefile    # Add write for others
chmod ugo-w somefile  # Remove write for all
chmod ug+x somefile   # Add execute for user and group
chmod g=rw somefile   # Set group to rw only
```

✅ 2. Octal Permission Table

Permission	Binary	Octal
---	000	0
--x	001	1
-w-	010	2
-wx	011	3
r--	100	4
r-x	101	5
rw-	110	6
rwx	111	7

✅ 3. Assign Full Permissions (Octal Style)

bash

```bash
chmod 764 somefile
# r-x rw- r--
# u   g   o

chmod 444 somefile
# r-- r-- r--  (read only to all)

chmod 555 somefile
# r-x r-x r-x  (read + execute to all)
```

📝 Quick Summary for Revision

Task	Command
Add write to others	`chmod o+w somefile`
Remove write from all	`chmod ugo-w somefile`
Add execute to user & group	`chmod ug+x somefile`
Set group as read and write (remove x)	`chmod g=rw somefile`
Set full access for owner, limited for others	`chmod 764 somefile`
Set read-only for all	`chmod 444 somefile`
Set read+execute for all	`chmod 555 somefile`

Let me know if you want:

- 📄 **PDF version**
- 🧪 **Practice questions**
- 📁 **Folder/File setup for testing**
  I'll prepare it quickly.

make neat and clean pdf of all of above

Your neat and clean PDF has been generated successfully.

📄 **Download it here:**
👉 Shell_Scripting_File_Permissions.pdf

Let me know if you want more such notes or practice sets!

i want a neat clean all information of this whole chat ....

It seems like I can’t do more advanced data analysis right now. Please try again later.

I wasn’t able to generate the full PDF summary of this chat at the moment. However, if you'd like, I can give you the **entire content as text** right here — which you can then copy into any editor (like MS Word or Notepad++) and convert it to a PDF yourself.

Would you like me to paste the full combined content now?

e entire content as text --yes give

Sure! Here's the **entire content of this chat**, neatly organized and formatted for your notes. You can copy and paste this into any text editor and save it as a PDF or DOC file.

✅ Shell Scripting & File Permissions – Full Notes

🔹 Ex1 – Print User Info Using Environment Variables

Script:

```bash
#!/bin/bash
echo "Home folder: $HOME"
echo "Current user: $USER"
echo "Current shell: $SHELL"
```

Environment Variables:

- `$HOME` → home directory of current user
- `$USER` → current username
- `$SHELL` → current shell path

🔹 Ex2 – PS1 and PS2, Escape Characters, Prompt Format

Script:

```bash
#!/bin/bash
echo "Value of PS1: $PS1"
echo "Value of PS2: $PS2"
```

Escape Characters:

Escape	Meaning
`\n`	New Line
`\t`	Tab
`\$`	Show `$` symbol
`$VAR`	Show var value

PS1 Format Example:

```bash
PS1="[\u@\h \W]\\$"
```

- `\u` = username
- `\h` = hostname
- `\W` = current folder
- `\\$` = show `$` symbol

Change Prompt Temporarily:

```bash
export PS1="Hi > "
```

🔹 Ex2 – File Permissions & `chmod`

Understanding `ls -l` Output:

```lua
rw-rw-r--
```

This format is:

```pgsql
OWNER | GROUP | OTHERS
 u  |  g  |  o
```

Each section:

- `r` = read
- `w` = write
- `x` = execute

🔸 Permissions in Context

For Files:

- `r` → view with `cat`, `vi`, etc.
- `w` → edit and save using editors
- `x` → run as program `./file`

For Folders:

- `r` → view content (`ls`)
- `w` → create/delete files (`touch`, `rm`)
- `x` → access folder (`cd`)

🔸 `chmod` – Change File/Folder Permissions

Add/Remove Permissions:

```bash
chmod o+w somefile        # Add write to others
chmod ugo-w somefile      # Remove write from all
chmod ug+x somefile       # Add execute to user and group
chmod g=rw somefile       # Set group as read+write only
```

Octal Table:

r w x	Binary	Octal
---	000	0
--x	001	1
-w-	010	2
-wx	011	3
r--	100	4
r-x	101	5
rw-	110	6
rwx	111	7

Assign Permissions Using Octal:

```bash
chmod 764 somefile
# rwx rw- r--

chmod 444 somefile
# r-- r-- r--
```

```
chmod 555 somefile
# r-x r-x r-x
```

🔹 Bonus Examples

✅ Ex1 – Print "Hello World"

```bash
#!/bin/bash
echo "Hello World"
```

✅ Ex2 – Print Contents of Folder

```bash
#!/bin/bash
echo "Enter path of folder:"
read fp
echo "The contents of $fp are:"
ls $fp
```

✅ Ex3 – Create Folder with Name Entered by User

```bash
#!/bin/bash
echo "Enter folder name:"
read foldername
mkdir $foldername
echo "Folder $foldername created."
```

✅ Ex4 – Create 0KB File in Parent Folder and Show Contents

```bash
#!/bin/bash
echo "Enter file name:"
read filename
touch ../$filename
echo "File created in parent folder."
echo "Contents of parent folder:"
ls ..
```

✅ Ex5 – Rename a File

```bash
#!/bin/bash
echo "Enter current file name:"
read oldname
echo "Enter new file name:"
read newname
mv $oldname $newname
echo "File renamed to $newname"
```

📌 Let me know if you want:

- PDF version (when the tool works)
- More exercises
- Viva questions based on these

All the best for your practice! 💻📁