Department of Computer Science & Engineering (Artificial Intelligence & Machine Learning) and Computer Science & Engineering (Cyber Security)

Unix Shell Programming Laboratory (CIL37/CYL37)

Lab Manual
(AY: 2024-25)

Prepared by:
Mrs.Shobha.K

Table of Contents

form month, day, and year. The script should check the validity of the argument and in the case of error, display a suitable message. **5.b.** Develop an awk script to delete duplicated line from a text file. The order of the original lines must remain unchanged.	
Program 6a. Type the below given text and save the file as 1.txt using Vi/Vim editor. Perform the below given operations. "Python is a very popular language. Python is easy to use. Python is easy to learn. Python is a cross-platform language HTML is a Markup Language Python Programming Language C Programming Language Shell Programming Perl Programming Language Bash" i. Replace all instances of a Python in a second line of 1.txt with Perl. ii. Replace the last occurrence of Programming with Scripting only of a match, not other instances. iii. Create a text file in the path /MSRIT/CSE/UG/Python.txt. Replace full path with just the filename no directory (such as Python.txt) and display it on standard output. iv. Add string before and after the matching pattern using '\1'. In the above given text, navigate yourself to the last line, you can find Bash keyword, Add Learn before Bash and Programming after Bash keyword. **6.b.** Perform the following execution using find command i. Find all the files in a current directory, whose permissions are 0777. ii. Assign a sticky bit to all the files in a current directory. iii. Find Directories with full permissions (777) and by	**20**

Vi/Vim editor operations:

1. Introduction to Unix and Shelling Programming, Vi/Vim
2. editor: Consider the following content and edit using Vi/Vim editor:

"The simplest way to understand how AI and ML relate to each other is: AI is the broader concept of enabling a machine or system to sense, reason, act, or adapt like a human. ML is an application of AI that allows machines to extract knowledge from data and learn from it autonomously.
Artificial Learning is the capability of a computer system to mimic human cognitive functions such as learning and problem-solving. Through AI, a computer system uses math and logic to simulate the reasoning that people use to learn from new information and make decisions
Cyber security is primarily about people, processes, and technologies working together to encompass the full range of threat reduction, vulnerability reduction, deterrence, international engagement, incident response, resiliency, and recovery policies and activities, including computer network operations, information assurance, law enforcement, etc."

a. Execute the following operations:
Open the file in vi/vim editor and type the above given content and save the file by name test1.txt, continue working in the opened file.
Navigate through the 10th line and go to save mode and write the 10th line to another file named test2.txt and continue working with test1.txt
Find the Keyword AI and replace AI with Artificial Learning using interactive substitution and save the current file.
Delete 5th line to a buffer (a), save the current file. Now switch to the file test2.txt, move to a desired location and copy the text. Toggle between previous file and current file.
Navigate to the first line in test.txt file, and go to 15th character towards right and move 3 lines down and 2 characters left and replace the character with M and save the current file and exit.

In the test2.txt file, move to 3 words forward in the current line and take the cursor to 2 words back, delete the word in the cursor position and navigate yourself to move to the line extremes, save and escape to the shell.
In the test1.txt, join first 3 lines (Join the current line with 2 lines).
Delete the 7th line in current file and exit the vi editor by saving and quitting.

Step1: To open the terminal
open terminal type-> vi test1.txt and inside test1.txt type the above text

Step2: To save a file
Save the test1.txt by pressing<ESC>and :wq ->to save the file Step3: To go to a particular line number
Press <Esc> and :5w test2.txt -> to copy the 5h line from test1.txt to test2.txt file

Step4:To replace with other word

Type :1,$ s/AI/Artificial Learning/gc→to replace AI with Artificial Learning in test1.txt file

It prompts **replace with fprintf (y/n/a/q/^E/^Y)?** to interactively substitute output will be displayed with AI replaced by Artificial Learning

Step5: To go to the particular line number vi test1.txt

Press<ESC> and type 5G->to goto the line number 5

Step6: To Delete:

Press together<ESC><shift> + <”> then type <add>
This will delete the text in the current line at where the cursor is positioning and stored into buffer. “add command (Current line will be deleted and stored in register a).

Step7:Go the beginning line number Type <1G>
Step8: To move 15 characters right from the current position
Type <15l>. This moves the cursor from the current position to the 15th character , which include white spaces too
Step 9: To go to 3rd line from the current position Type <3j>

Step 10: To Move one character left Press<ESC> Type <h>

Step 11: To Move one character right Press<ESC> Type <l>

Step 12: To replace the current character
Press <ESC> type <r>→We can replace it with another character
Type <:w>(To save file)

Step 13: To open a file
Type <e test2.txt> . To go to file test2.txt

Step 14: To move the cursor to 3 words forward Type <3e>

Step 15: To move 2 words backward Type <2b>

Step 16: Join the next line with the current line position.
Type <2J>

b. Correct the below given c program in Vi/Vim editor using vi commands.

Sample.c(Before Correction—with errors)	Sample.c(After Correction--Right)

#include<stio.h> #include errno.h
int test(int * message)
{print("Errno is %8d',errno)
exit;
	#include <stdio.h>
	#include <stdlib.h> // Required for exit() #include <errno.h>
	void test(char *message)
	{
	printf("Errno is %8d\n", errno);
	// Print the value of errno
	exit(1); // Exit the program with status 1
	}
	int main()
	{
	test("Some message");
	// Call the test function return 0;
	// Indicate successful termination
	}

Step1:vi sample.c

Type the corresponding program, which is mentioned at the left side (Before Correction—with errors)

Press <Esc> and:wq.Program will be saved. For correcting the program give vi sample.c
Using the commands which you done in Ist experiment edit accordingly

OUTPUT:

```
rit-admin@ritadmin:~$ vi sample.c
 rit-admin@ritadmin:~$ cc sample.c
rit-admin@ritadmin:~$ ./a.out Errno is            0
```

Shell Scripting

2. Consider the below scenarios and execute the given shell scripts.

"Ramaiah College is having 10 departments (Say, CS, IS, AI, ML, Cyber Security, EC, Mechanical, EEE, DS, Civil) with UG and PG programs, and in each of the program student details, course details are maintained in 10 different files (such as Student Details, Course details, Curriculum, Exam, Marks, Research Activity, NBA, Placement Activities, Library Details, Extra Activities….)."

a.Develop a shell script for the above scenario to create 10 levels of folders for the departments and inside each level(department) of folder, create 10 files in each department for maintaining student details. Display the entire hierarchy on the standard output by using tree command.

Note: Commands To be used are: mkdir,find and touch
for loop to be used.
Step1:vi prog2a.sh

```
#!/bin/sh


# Create directories from 1 to 10 without quotes
```

```
for i in $(seq 1 10);
 do
    mkdir "$i"
     done

# Create files named name_date.txt-001 to name_date.txt-010 in each directory for
dir in $(find . -maxdepth 1 -type d);
do
for j in $(seq -f "%03g" 1 10);
 do
 touch "$dir/name_date.txt-$j"
       done
       done
```

OUTPUT

```
rit-admin@ritadmin:~$ mkdir rit
rit-admin@ritadmin:~$ cd rit
rit-admin@ritadmin:~/rit3$ vi prog2a.sh
 rit-admin@ritadmin:~/rit3$ sh prog2.sh


 rit-admin@ritadmin:~/rit3$ ls
1    5  name_date.txt-005  name_date.txt-010
10  6  name_date.txt-001  name_date.txt-006  pgm.sh
2    7  name_date.txt-002  name_date.txt-007
3    8  name_date.txt-003  name_date.txt-008
4    9  name_date.txt-004  name_date.txt-009


rit-admin@ritadmin:~/rit2$ tree
.
├── 1
│   ├── name_date.txt-001
│   ├── name_date.txt-002
│   ├── name_date.txt-003
│   ├── name_date.txt-004
│   ├── name_date.txt-005
│   ├── name_date.txt-006
│   ├── name_date.txt-007
│   ├── name_date.txt-008
│   ├── name_date.txt-009
│   └── name_date.txt-010
├── 10
│   ├── name_date.txt-001
│   ├── name_date.txt-002
│   ├── name_date.txt-003
│   ├── name_date.txt-004
│   ├── name_date.txt-005
│   ├── name_date.txt-006
│   ├── name_date.txt-007
│   ├── name_date.txt-008
│   ├── name_date.txt-009
│   └── name_date.txt-010
├── 2
│   ├── name_date.txt-001
│   ├── name_date.txt-002
│   ├── name_date.txt-003
│   ├── name_date.txt-004
│   ├── name_date.txt-007
│   ├── name_date.txt-008
│   ├── name_date.txt-009
│   └── name_date.txt-010
├── 4
│   ├── name_date.txt-001
│   ├── name_date.txt-002
│   ├── name_date.txt-003
│   ├── name_date.txt-004
│   ├── name_date.txt-005
│   ├── name_date.txt-006
│   ├── name_date.txt-007
│   ├── name_date.txt-008
│   ├── name_date.txt-009
│   └── name_date.txt-010
├── 5
│   ├── name_date.txt-001
│   ├── name_date.txt-002
│   ├── name_date.txt-003
│    ├── name_date.txt-004
│    ├── name_date.txt-005
│    ├── name_date.txt-006
│    ├── name_date.txt-007
│    ├── name_date.txt-008
│    ├── name_date.txt-009
│    └── name_date.txt-010
├── 6
│    ├── name_date.txt-001
│    ├── name_date.txt-002
│    ├── name_date.txt-003
│    ├── name_date.txt-004
│    ├── name_date.txt-005
│    ├── name_date.txt-006
│    ├── name_date.txt-007
│    ├── name_date.txt-008
│    ├── name_date.txt-009
│    └── name_date.txt-010
├── 7
│    ├── name_date.txt-001
│    ├── name_date.txt-002
│    ├── name_date.txt-003
│    ├── name_date.txt-004
│    ├── name_date.txt-005
│    ├── name_date.txt-006
│    ├── name_date.txt-007
│    ├── name_date.txt-008

│    ├── name_date.txt-009
│    └── name_date.txt-010
├── 8
│    ├── name_date.txt-001
│    ├── name_date.txt-002
│    ├── name_date.txt-003
│    ├── name_date.txt-004
│    ├── name_date.txt-005
│    ├── name_date.txt-006
│    ├── name_date.txt-007
│    ├── name_date.txt-008
│    ├── name_date.txt-009
│    └── name_date.txt-010
```

```
├─ 9
│    ├──── name_date.txt-001
│    ├──── name_date.txt-002
│    ├──── name_date.txt-003
│    ├──── name_date.txt-004
│    ├──── name_date.txt-005
│    ├──── name_date.txt-006
│    ├──── name_date.txt-007
│    ├──── name_date.txt-008
│    ├──── name_date.txt-009
│    └──── name_date.txt-010
```

a) Develop a shell script that accepts above created filename as argument and display its creation time and permissions of the file, on the standard output.

Step1:vi prog2b.sh

Note: Commands used: echo,ls,cut
If condition used for checking file name exists

```
#/bin/bash
if [ $# -eq 0 ];
then
echo "display does not exit"
 else
ls -l $1 > t2
x=`cut -d ' ' -f 1,6,7,8,9 t2`
 echo $x
fi
```

Output:-

vi prog2b.sh // Type your above pgm
sh **prog2b.sh** t2 // To compile it Output shown as:

```
-rw-rw-r-- Sep 24 11:43 t2
```

3. a. Develop a shell script that takes a valid directory name as an argument and recursively descend all the sub-directories, finds the maximum length of any file in that hierarchy, and store the output in a file.
Step1:vi prog3a.sh Shell Script:
Note: Commands Used:echo,ls,cut,tr,tail
For loop and if condition used

```
#/bin/bash
for i in $*
 do
if [ -d $i ];
then
echo "large filename size is"
  echo `ls -Rl $1 | grep "^-" | tr -s ' ' | cut -d' ' -f 5,8 | sort -n| tail -1`
   else
 echo "not directory"
  fi
done
```

Output:

```
sh prog3a.sh cy            # cy is already existing directory name. large filename size is
135 11:39
```

3.b.Create a shell script to find a file with particular name, (show separate outputs for both the conditions)
if that file exists then rename the existing file and create an empty file with that name.
if that file does not exist then create a new empty file.
If both the conditions are done together.

Note:Commands used :echo,mv,touch

If condition used to check existing file

Step1: vi prog3b.sh

```
#/bin/bash
# Set the desired file name
file_name="example.txt"

# Check if the file exists
if [ -e "$file_name" ];
 then
# If the file exists, rename it and create an empty file with the original name
 echo "File '$file_name' exists."

# Rename the existing file
mv "$file_name" "${file_name}_old"
echo "Existing file renamed to '${file_name}_old'."

# Create an empty file with the original name
touch "$file_name"
echo "New empty file created with the name '$file_name'."

echo "Both conditions done together."
else
# If the file does not exist, create a new empty file
touch "$file_name"
echo "File '$file_name' does not exist. New empty file created."
fi

# Additional message to indicate script completion
echo "Script execution completed."
```

Output:

```
exam@ThinkCentre- M70t:~$ vi prog3.sh exam@ThinkCentre- M70t:~$ sh prog3.sh
File 'example.txt' does not exist. New empty file created. Script execution completed.
exam@ThinkCentre-M70t:~$ ls                              # This will show example.txt
```

```
filename
exam@ThinkCentre-M70t:~$ cat > example.txt                              # to enter the
content inside it
hello good


Ctrl+Z (To terminate) exam@ThinkCentre-M70t:~$ cat example.txt
hello good
exam@ThinkCentre-M70t:~$ sh prog3.sh File 'example.txt' exists.
Existing file renamed to 'example.txt_old'.
New empty file created with the name 'example.txt'.
Both conditions done together.
Script execution completed.


exam@ThinkCentre-M70t:~$ ls # This will show example.txt_old filename by
replacing example.txt
exam@ThinkCentre-M70t:~$ cat example.txt_old               # To see the content
hello
good
```

4.a. Build a shell script to display the system space used. If it is greater than 80%, display as Low System Space and list the files having size greater than 1GB.

Note:commands used :awk,echo,sed

Step1:vi prog4a.sh

```
#!/bin/bash

# Get system space usage percentage

space_usage=$(df -h / | awk 'NR==2 {print $5}' | sed 's/%//')

echo "System Space Usage: $space_usage%"

# Check if system space usage is greater than 80%

 if [ "$space_usage" -gt 80 ];

 then
```

```
  echo "Low System Space"
 # List files greater than 1GB echo "Files larger than 1GB:"
  find / -type f -size +1G -exec ls -lh {} \; 2>/dev/null



# Check if find command found anything
 if [ $? -ne 0 ];
 then
        echo "No files larger than 1GB found or insufficient permissions."
     fi
     else
  echo "System space usage is under control."
  fi
```

Output:

```
rit-admin@ritadmin:~$ sh prog4a.sh System Space Usage: 24
System space usage is under control.
rit-admin@ritadmin:~$
```

b. Write a shell program to count the number of words, characters, white spaces,

and special symbols in each text and display the output on standard output.

Commands Used:echo,wc

Step1:vi prog4b.sh

```
#!/bin/bash
# Function to count characters, words, white spaces, and special symbols
count_text_stats() {
input_text="$1"


# Count characters
char_count=$(echo -n "$input_text" | wc -m)

# Count words
word_count=$(echo "$input_text" | wc -w)

# Count white spaces
space_count=$(echo "$input_text" | tr -cd ' ' | wc -c)

# Count special symbols
special_count=$(echo "$input_text" | tr -cd '[:punct:]' | wc -c)

# Print analysis report
echo "Text Analysis Report:" echo " "
echo "Character count: $char_count"
 echo "Word count: $word_count"
echo "White space count: $space_count"
echo "Special symbol count: $special_count" echo " "
}

# Example text (you can replace this with your own text or input)
text_to_analyze="This is an example text! It contains special symbols, such as @ and
#."

# Call the function to analyze the text
 count_text_stats "$text_to_analyze"

# Additional message to indicate script completion echo "Script execution
completed."
```

Output

```
exam@ritadmin:~$ vi prog4b.sh
exam@ritadmin:~$ sh prog4b.sh Text Analysis Report:

Character count: 70
Word count: 14
White space count: 13 Special symbol count: 5

Script execution completed.
```

5.a. Develop an awk script that accepts date arguments in the form of dd-mm-yy and display it in the form month, day, and year. The script should check the validity of the argument and in the case of error, display a suitable message.

Step1:prog5a.sh Code:

5.a. Develop an awk script that accepts date arguments in the form of dd-mm-yy and display it in the form month, day, and year. The script should check the validity of the argument and in the case of error, display a suitable message.

Step1:prog5a.sh Code:

```
rit-admin@ritadmin:~$ vi pgr5.awk


#!/usr/bin/awk -f

{
   split($0, arr, "-") # Split input by "-" into the arr array
   day = arr[1]
   month = arr[2]
   year = arr[3]

   # Validate day and month
   if (day < 1 || day > 31 || month < 1 || month > 12) {
      print "Invalid date"
      exit 0
   }

   # Print the day
   print day

   # Print month name based on the value of month
   if (month == 1) {
      print "Jan"
   } else if (month == 2) {
        print "Feb"
    } else if (month == 3) {
        print "March"
    } else if (month == 4) {
        print "April"
    } else if (month == 5) {
        print "May"
    } else if (month == 6) {
        print "Jun"
    } else if (month == 7) {
        print "Jul"
    } else if (month == 8) {
        print "Aug"
    } else if (month == 9) {
        print "Sep"
    } else if (month == 10) {
        print "Oct"
    } else if (month == 11) {
        print "Nov"
    } else if (month == 12) {
        print "Dec"
    }

    # Print the year
    print year
}
```

output:

```
rit-admin@ritadmin:~$ awk -f pgr5.awk

04-09-2024

04

Sep

2024
```

b.Develop an awk script to delete duplicated line from a text file. The order of the original lines must remain unchanged.

Step1:vi prog5b.awk

Code:

```
#!/usr/bin/awk -f
BEGIN {
    print "Removing duplicated lines..."
    no = 0
}

{
    line[++no] = $0  # Store the current line in the array and increment the counter
   }
END {
    for (i = 1; i <= no; i++) {
        flag = 1  # Assume the line is unique
        for (j = 1; j < i; j++) {
            if (line[i] == line[j]) {
                flag = 0  # Found a duplicate
                break;    # No need to check further
            }
        }
        if (flag == 1) {
            print line[i] >> "out13a.txt"  # Print unique line to the output file
        }
    }
}
```

Output:

```
     exam@h-primary:~$ vi prog5b.awk
exam@h-primary:~$ >out13a.txt // To remove all the content in the file “out13a.txt”
  exam@h-primary:~$ cat out13a.txt
 exam@h-primary:~$ cat input.txt // To add content in the input file
 mat
cat
```

```
 pat
  cat
 mat
   bat
  exam@h-primary:~$ awk -f prog5b.awk input.txt // To compile
Removing duplicated lines...
 exam@h-primary:~$ cat out13a.txt // To display after removing duplicate content
   mat
   cat
 pat
 bat
```

OR
Through AWK command

awk '!seen[$0]++' in.txt > out.txt

OUTPUT

```
exam@h-primary:~$ cat > in.txt
 orange
 apple
mango
apple
 orange
 peer
  ^Z
    [1]+  Stopped                 cat > in.txt
exam@h-primary:~$ awk '!seen[$0]++' in.txt > out.txt
  exam@h-primary:~$ cat out.txt
orange
apple
mango
  peer
  exam@h-primary:~$ cat in.txt
 orange
  apple
 mango
 apple
```

orange
peer

6.a. Type the below given text and save the file as 1.txt using Vi/Vim editor. Perform the below given operations.
"Python is a very popular language.
Python is easy to use. Python is easy to learn. Python is a cross-platform language
HTML is a Markup Language Python Programming Language C Programming
Language Shell Programming Perl Programming Language Bash"

i. Replace all instances of a Python in a second line of 1.txt with Perl.
ii. Replace the last occurrence of Programming with Scripting only of a match, not other instances.

iii. Create a text file in the path /MSRIT/CSE/UG/Python.txt. Replace full path with just the filename no directory (such as Python.txt) and display it on standard output.
iv. Add string before and after the matching pattern using '\1'. In the above given

text, navigate yourself to the last line, you can find Bash keyword, Add Learn before Bash and Programming after Bash keyword.

Steps or Solution or Output: exam@ThinkCentre-M70t:~$ cat >1.txt

```
"Python is a very popular language.
Python is easy to use. Python is easy to learn. Python is a cross-platform language
HTML is a Markup Language Python Programming Language C Programming
    Language Shell Programming Perl Programming Language
Perl Programming Language Bash"

exam@ThinkCentre-M70t:~$ cat 1.txt
"Python is a very popular language.
Python is easy to use. Python is easy to learn. Python is a cross-platform language
HTML is a Markup Language Python Programming Language C Programming
    Language Shell Programming Perl Programming Language
Perl Programming Language Bash"
exam@ThinkCentre-M70t:~$ sed '1,9 s/Python/perl/g' 1.txt
 "perl is a very popular language. perl is easy to use. perl is easy to learn. perl is a
```

cross-platform language HTML is a Markup Language
perl Programming Language C Programming Language Shell Programming Perl Programming Language Perl Programming Language Bash"

b. Perform the following execution using find command
i.Find all the files in a current directory, whose permissions are 0777.
ii.Assign a sticky bit to all the files in a current directory.
Iii.Find Directories with full permissions (777) and by using chmod command change the permissions by assigning read, write and execute permissions to owner and only read & execute to group and others.
v.Find last 20 days modified files, accessed files.

v.Find all the files which are modified in last 1 hour.

```
rit-admin@ritadmin:~$ find . -type f -perm 0777
./rit2/pgm.sh
./cy/66.txt

./cy/pgm2b.sh
rit-admin@ritadmin:~$
```

ii.   Assign a sticky bit to all the files in the current directory:

```
find . -type f -exec chmod +t {} +
```

Output:

```
rit-admin@ritadmin:~$ sudo find . -type f -exec chmod +t {} +
[sudo] password for rit-admin:
rit-admin@ritadmin:~$ ls -l ./Bigdata/hadoop- 3.2.2/bash.sh
-rw-r--r-T 1 root root 158 Feb 29  2024
./Bigdata/hadoop-3.2.2/bash.sh
```

iii. Find directories with full permissions (777) and change the permissions as described:

```
find . -type d -perm 777 -exec chmod 755 {} +
```

Output:

```
rit-admin@ritadmin:~$ find . -type d -perm 777 - exec chmod 755 {} +
```

iv. Find files modified in the last 20 days and files accessed in the last 20 days:

```
find . -type f -mtime -20
```

Output will be displayed with 20 days data

v.Find all files modified in the last 1 hour:

```
find . -type f -mmin -60
```

Output will be displayed with last 1 hour

C Programs by using UNIX File System Calls

7. a. Develop a C program to emulate the UNIX ls –li command, which lists all the attributes of the files in a specified directory.

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <dirent.h>
#include <unistd.h>
#include <pwd.h>
#include <grp.h>
#include <time.h>

// Function to print file permissions
void print_permissions(struct stat fileStat) {
    printf((S_ISDIR(fileStat.st_mode)) ? "d" : "-");
    printf((fileStat.st_mode & S_IRUSR) ? "r" : "-");
    printf((fileStat.st_mode & S_IWUSR) ? "w" : "-");
    printf((fileStat.st_mode & S_IXUSR) ? "x" : "-");
    printf((fileStat.st_mode & S_IRGRP) ? "r" : "-");
    printf((fileStat.st_mode & S_IWGRP) ? "w" : "-");
    printf((fileStat.st_mode & S_IXGRP) ? "x" : "-");
    printf((fileStat.st_mode & S_IROTH) ? "r" : "-");
    printf((fileStat.st_mode & S_IWOTH) ? "w" : "-");
    printf((fileStat.st_mode & S_IXOTH) ? "x" : "-");
}

// Function to print file information similar to `ls -li`
void list_directory(const char *dirpath) {
    DIR *dir;
    struct dirent *entry;
    struct stat fileStat;
    char fullpath[1024];

    // Open directory
    dir = opendir(dirpath);
    if (dir == NULL) {
        perror("opendir");
        exit(EXIT_FAILURE);
    }

    // Read directory entries
    while ((entry = readdir(dir)) != NULL) {
        snprintf(fullpath, sizeof(fullpath), "%s/%s", dirpath, entry->d_name);

        // Get file statistics
        if (stat(fullpath, &fileStat) == -1) {
            perror("stat");
            continue;
        }

        // Print inode number
        printf("%ld ", (long)fileStat.st_ino);

        // Print file permissions
        print_permissions(fileStat);

        // Print number of links
        printf(" %ld ", (long)fileStat.st_nlink);

        // Print owner and group
        printf("%s %s ", getpwuid(fileStat.st_uid)->pw_name, getgrgid(fileStat.st_gid)-
>gr_name);

        // Print file size
        printf("%5ld ", (long)fileStat.st_size);

        // Print last modification time
        char timebuf[80];
        struct tm *timeinfo = localtime(&fileStat.st_mtime);
        strftime(timebuf, sizeof(timebuf), "%b %d %H:%M", timeinfo);
        printf("%s ", timebuf);

        // Print file name
        printf("%s\n", entry->d_name);
    }

    // Close directory
    closedir(dir);
}

int main(int argc, char *argv[]) {
    const char *dirpath;

    // Check if directory path is provided as an argument
    if (argc > 1) {
        dirpath = argv[1];
    } else {
        // Default to current directory if no argument provided
        dirpath = ".";
    }

    list_directory(dirpath);

    return 0;
}
```

Output:

```
 rit-admin@ritadmin:~$ vi prog7.c
rit-admin@ritadmin:~$ vi program7.c
rit-admin@ritadmin:~$ cc program7.c
 rit-admin@ritadmin:~$ ./a.out
 6074738 -rw-rw-r-- 1 rit-admin rit-admin    12 Sep 25 14:11 create.sh
 6073890 -rw-rw-r-- 1 rit-admin rit-admin    39 Sep 24 13:05 prog3c.sh

  6072344 -rw-rw-r-- 1 rit-admin rit-admin   288 Sep 24 10:24 test2.c
  6075492 -rw-rw-r-- 1 rit-admin rit-admin   549 Sep 25 15:17 1.txt

       6029383 drwxr-xr-x 2 rit-admin rit-admin  4096 Jun 03 12:44 Documents
        6029315 -rw-r--r-- 1 rit-admin rit-admin   807 Feb 28 17:40 .profile
      6074118 -rw-rw-r-- 1 rit-admin rit-admin   645 Sep 25 14:43 prgm5a.sh
      6075415 -rw-rw-r-- 1 rit-admin rit-admin   516 Sep 25 15:00    pnprogram5.awk
     6075645 -rw-rw-r-- 1 rit-admin rit-admin  1438 Sep 25 15:35 prog7.c
     6029321 drwx------ 14 rit-admin rit-admin  4096 May 27 10:58 .config
     6029856 -rw------- 1 rit-admin rit-admin    18 Feb 28 18:06 .mysql_history
     6073205 -rw-rw-r-- 1 rit-admin rit-admin   190 Sep 24 11:51 prog3.sh
    6034208 -rw-rw-r-- 1 rit-admin rit-admin   565 Sep 25 14:56 nprog5a.sh
```

7.b. Write a C program to remove empty files from the given directory using system
calls.

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <dirent.h>
#include <sys/stat.h>
 #include <unistd.h>
void removeEmptyFiles(char *path)
{
DIR *dir;
struct dirent *entry; struct stat fileStat;
// Open the directory
if ((dir = opendir(path)) == NULL) {

perror("Error opening directory"); exit(EXIT_FAILURE);
}
// Read each entry in the directory while ((entry = readdir(dir)) != NULL) {



// Construct the full path char filePath[1024];
snprintf(filePath, sizeof(filePath), "%s/%s", path, entry->d_name);
// Get file status

if (stat(filePath, &fileStat) < 0)
 {
 perror("Error getting file status");
 exit(EXIT_FAILURE);
}
// Check if the file is empty and remove it
if (S_ISREG(fileStat.st_mode) && fileStat.st_size == 0)
{
if (unlink(filePath) == 0)
{
printf("Removed empty file: %s\n", entry->d_name);
}
Else
```

```
 {
perror("Error removing file");
}
}
}
// Close the directory closedir(dir);
}
int main(int argc, char *argv[]) {
// Check if the correct number of arguments is provided
 if (argc != 2)

 {
fprintf(stderr, "Usage: %s <directory>\n", argv[0]);
 exit(EXIT_FAILURE);
}


// Call the function to remove empty files
removeEmptyFiles(argv[1]);
return 0;
}
```

Compilation Steps:
Assuming you save the program in a file named remove_empty_files.c, open a terminal and follow
these steps:

1. gcc remove_empty_files.c -o remove_empty_files
   This command uses the GCC compiler to compile the C program and produces an executable named
   remove_empty_files.

2. ./remove_empty_files /path/to/directory
   Replace /path/to/directory with the actual path of the directory you want to process.

Output:

```
rit-admin@ritadmin:~$ vi prog7b.c
rit-admin@ritadmin:~$ vi progn7b.c
rit-admin@ritadmin:~$ cc progn7b.c
rit-admin@ritadmin:~$ ./a.out .
Removed empty file: .sudo_as_admin_successful
Removed empty file: .scala_history
Removed empty file: example.txt
rit-admin@ritadmin:~$
```

Explanation:

The removeEmptyFiles function opens the specified directory, reads each entry, checks if it's
a regular file with zero size, and removes it using the unlink system call.

The main function checks if the correct number of command-line arguments is provided, then
calls removeEmptyFiles with the specified directory path.

Make sure to run the program with proper permissions to delete files in the specified directory.

Program 8.a. Write a C program to read n characters from a file and append them back to the same file using dup2 function.

Step1:vi prog8a.c

```
 #include <stdio.h>

 #include <stdlib.h>

#include <unistd.h>

#include <fcntl.h>

#define MAX_SIZE 100
```

```
 int main() {
char filename[] = "file.txt";
int fd, new_fd;
char buffer[MAX_SIZE];
ssize_t bytes_read;
int n;
printf("Enter the number of characters to read and append: ");
scanf("%d", &n);
// Open the file
fd = open(filename, O_RDWR);
if (fd == -1) {
perror("Error opening file");
exit(EXIT_FAILURE);
}
// Move file pointer to the end
 lseek(fd, 0, SEEK_END);
// Duplicate file descriptor
new_fd = dup(fd);
if (new_fd == -1) {
perror("Error duplicating file descriptor");
 close(fd);
exit(EXIT_FAILURE);
}
```

```
// Move the file pointer back 'n' characters
lseek(fd, -n, SEEK_END);
// Read 'n' characters from the file
bytes_read = read(fd, buffer, n);
if (bytes_read == -1)
{
perror("Error reading file");
 close(fd);
close(new_fd);
exit(EXIT_FAILURE);
}
// Append the read characters back to the file
if (write(new_fd, buffer, bytes_read) != bytes_read)
 {
 perror("Error writing to file");
close(fd);
close(new_fd);
exit(EXIT_FAILURE);
}
// Close file descriptors

close(fd);
close(new_fd);
printf("%d characters read and appended successfully.\n", (int)bytes_read);
```

```
return 0;

}
```

Output:

```
rit-admin@ritadmin:~$ vi prog8a.c

rit-admin@ritadmin:~$ vi progn8a.c

rit-admin@ritadmin:~$ cc progn8a.c

rit-admin@ritadmin:~$ ./a.out

Enter the number of characters to read and append: 2

2 characters read and appended successfully.
```

8.b. Write a C program to list all files in a current directory using system calls.

vi prog8b.c

```
#include <stdio.h>

#include <stdlib.h>

#include <dirent.h>



// Function to list files in the specified directory

void list_files(const char *dirpath) {

    DIR *dir;

    struct dirent *entry;



    // Open the directory
    dir = opendir(dirpath);
    if (dir == NULL) {
        perror("Error opening directory");
        exit(EXIT_FAILURE);
    }

    // Read entries in the directory
    while ((entry = readdir(dir)) != NULL) {
        // Skip the current (.) and parent (..) directory entries
        if (entry->d_name[0] == '.') {
            continue; // Skipping hidden files and directories
        }
        printf("%s\n", entry->d_name); // Print the file name
    }

    // Close the directory
    closedir(dir);
}

int main() {
    const char *current_directory = "."; // Current directory
    printf("Files in the current directory:\n");
    list_files(current_directory);


    return 0;
}
```

Output:

```
rit-admin@ritadmin:~$ vi progn8b.c
rit-admin@ritadmin:~$ cc progn8b.c
rit-admin@ritadmin:~$ ./a.out
Files in the current directory:
prog8a.c
prog8nb.c
1.txt
```

9a. Create a C program to simulate the copy command in Unix (cp command)

```
#include <stdio.h>
#include <stdlib.h>
```

```
#define BUFFER_SIZE 1024
int main(int argc, char *argv[]) {
    if (argc != 3) {
        fprintf(stderr, "Usage: %s <source_file> <destination_file>\n", argv[0]);
        exit(EXIT_FAILURE);
    }

    FILE *source_file, *destination_file;
    char buffer[BUFFER_SIZE];
    size_t bytesRead;
    // Open the source file in binary read mode
    source_file = fopen(argv[1], "rb");
    if (source_file == NULL) {
        perror("Error opening source file");
        exit(EXIT_FAILURE);
    }

    // Open or create the destination file in binary write mode
    destination_file = fopen(argv[2], "wb");
    if (destination_file == NULL) {
        perror("Error opening destination file");
        fclose(source_file); // Close the source file if destination cannot be opened
        exit(EXIT_FAILURE);
    }

    // Copy the content from source to destination
    while ((bytesRead = fread(buffer, 1, BUFFER_SIZE, source_file)) > 0) {
        fwrite(buffer, 1, bytesRead, destination_file);
    }
    // Close the files
    fclose(source_file);
    fclose(destination_file);

    printf("File copied successfully.\n");
    return 0;
}
```

Output:

```
rit-admin@ritadmin:~$ vi pro9a.c
rit-admin@ritadmin:~$ cc pro9a.c
rit-admin@ritadmin:~$ ./a.out
rit-admin@ritadmin:~$ ./a.out input1.txt date.txt
```

```
File copied successfully.

rit-admin@ritadmin:~$ cat date.txt

hello

hello good
```

b. Develop a C program to simulate the ls (list) Unix command using system calls.

Step1:vi prog9b.c

```
 #include <stdio.h>
#include <stdlib.h>
#include <dirent.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <string.h>

void list_directory(const char *dirpath, int show_hidden) {
    DIR *dir;
    struct dirent *entry;

    // Open the specified directory
    dir = opendir(dirpath);
    if (dir == NULL) {
        perror("Error opening directory");
        exit(EXIT_FAILURE);
    }
    // Read and print all files and directories in the directory
    while ((entry = readdir(dir)) != NULL) {
        // Skip hidden files if specified
        if (!show_hidden && entry->d_name[0] == '.') {
            continue; // Skip entries that start with '.'
        }
        printf("%s\n", entry->d_name); // Print the name of each entry
    }

    // Close the directory
    closedir(dir);
}

int main(int argc, char *argv[]) {
    const char *dirpath = "."; // Default to current directory
    int show_hidden = 0;       // Flag to control showing hidden files

    // Check command-line arguments
    for (int i = 1; i < argc; i++) {
        if (strcmp(argv[i], "-a") == 0) {
            show_hidden = 1; // Show hidden files if -a flag is provided
        } else {
            dirpath = argv[i]; // Use the specified directory path
        }
    }

    // List the directory contents
    list_directory(dirpath, show_hidden);
    return 0;
}
```

Output:

```
rit-admin@ritadmin:~$ vi prognb.c

rit-admin@ritadmin:~$ cc prognb.c

rit-admin@ritadmin:~$ ./a.out

prog8a.c

prog8nb.c

1.txt

Documents

sample2.c

newa.awk

samp4.c

prog7.c

example.txt_old
```

Program 10a. Understanding File Descriptors and Building a C program to create two processes P1 and P2. P1 takes a string and passes it to P2. P2 concatenates the received string with another string without using string function and sends it back to P1 for printing, send the output to standard output.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>

#define BUFFER_SIZE 256

int main() {
    int pipe1[2]; // Pipe for P1 to P2
    int pipe2[2]; // Pipe for P2 to P1
```

```
pid_t pid;

// Create pipes
if (pipe(pipe1) == -1 || pipe(pipe2) == -1) {
    perror("Pipe creation failed");
    exit(EXIT_FAILURE);
}

// Create the child process
pid = fork();
if (pid < 0) {
    perror("Fork failed");
    exit(EXIT_FAILURE);
}

if (pid > 0) { // P1 process
    close(pipe1[0]); // Close read end of pipe1
    close(pipe2[1]); // Close write end of pipe2

    char input[BUFFER_SIZE];
    printf("Enter a string: ");
    fgets(input, BUFFER_SIZE, stdin);

    // Remove newline character from input
    input[strcspn(input, "\n")] = 0;

    // Send the string to P2
    write(pipe1[1], input, strlen(input) + 1);

    // Read concatenated string from P2
    char output[BUFFER_SIZE];
    read(pipe2[0], output, BUFFER_SIZE);
    printf("Concatenated String from P2: %s\n", output);

    close(pipe1[1]); // Close write end of pipe1
    close(pipe2[0]); // Close read end of pipe2
} else { // P2 process
    close(pipe1[1]); // Close write end of pipe1
    close(pipe2[0]); // Close read end of pipe2

    char received[BUFFER_SIZE];
```

```
        // Read the string from P1
        read(pipe1[0], received, BUFFER_SIZE);

        // Concatenate with another string
        const char *concat_str = " - Processed by P2";
        char result[BUFFER_SIZE];

        // Manual concatenation without using string functions
        int i = 0, j = 0;

        // Copy received string to result
        while (received[i] != '\0') {
            result[i] = received[i];
            i++;
        }

        // Copy concatenation string to result
        while (concat_str[j] != '\0') {
            result[i] = concat_str[j];
            i++;
            j++;
        }
        result[i] = '\0'; // Null terminate the result

        // Send the concatenated string back to P1
        write(pipe2[1], result, strlen(result) + 1);

        close(pipe1[0]); // Close read end of pipe1
        close(pipe2[1]); // Close write end of pipe2
        exit(0);
    }
    return 0;
}
```

Output:

```
rit-admin@ritadmin:~$ vi prog10.c

rit-admin@ritadmin:~$ cc prog10.c rit-admin@ritadmin:~$ ./a.out Enter a string:
Ramaiah


Concatenated String from P2: ramaiah - Processed by P2 rit-admin@ritadmin:~$
```

10.b.Create a C program to simulate Grep Unix Command using system calls.

Code:

```
#include <stdio.h>
 #include <stdlib.h>
 #include <fcntl.h>
 #include <unistd.h>
 #include <string.h>

#define BUFFER_SIZE 1024

void grep_pattern(const char *filename, const char *pattern)
 { int fd;
   char buffer[BUFFER_SIZE];
   ssize_t bytesRead;
   char *line;

   // Open the file for reading
   fd = open(filename, O_RDONLY);
   if (fd < 0) {
   perror("Error opening file");
   exit(EXIT_FAILURE);
   }



   // Read the file line by line
while ((bytesRead = read(fd, buffer, sizeof(buffer))) > 0)
 { buffer[bytesRead] = '\0'; // Null-terminate the buffer
   line = strtok(buffer, "\n"); // Tokenize the buffer to get each line
```

```
    while (line != NULL) {
if (strstr(line, pattern) != NULL) { // Check if the pattern exists in the line
    printf("%s\n", line); // Print the matching line
    }
    line = strtok(NULL, "\n"); // Move to the next line



    }
    }

if (bytesRead < 0)
{
perror("Error reading file");
    }

    // Close the file descriptor close(fd);
    }




int main(int argc, char *argv[])
{
 if (argc != 3) {
    fprintf(stderr, "Usage: %s <pattern> <filename>\n", argv[0]); exit(EXIT_FAILURE);
    }

    const char *pattern = argv[1];
    const char *filename = argv[2];

    // Call the grep function
    grep_pattern(filename, pattern);

    return 0;
    }
```

Output:

```
rit-admin@ritadmin:~$ vi prog10b.c rit-admin@ritadmin:~$ vi prm10b.c rit-
admin@ritadmin:~$ cc prm10b.c

rit-admin@ritadmin:~$ ./a.out Usage: ./a.out <pattern> <filename> rit-
admin@ritadmin:~$
```

XML Integration

Program11.Consider the student details given below and create an XML file and save it as sample.xml.

Student Details: -

Sl. No.	**Student Name**	**USN**	**Department Name**
1.	Alex	1RITCS001	CSE
2.	Smith	1RITDS040	DS
3.	Saliena	1RITCV051	Civil
4.	Elizabeth	1RITME011	Mechanical

Department Details: -

Sl. No.	**Department Name**	**Department Id**	**Total no of enrolled students**
1.	CSE	01	170
2.	DS	02	188
3.	Civil	03	160
4.	Mechanical	04	150

a) From the above XML file, separate only student details and redirect the output to a file.
b) Replacing the tag name from name to Dept name and change globally in a shell script.

:) Read the tag value of USN and redirect the output to standard output as well as redirect to a file.

Solution:

```
exam@ThinkCentre-M70t:~$ sudo apt-get install xml-twig-tools
exam@ThinkCentre-M70t:~$ sudo apt-get install xsltproc exam@ThinkCentre-
M70t:~$ gedit sample.xml
```

Source Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<university>
<students>

<student>

<serialNumber>1</serialNumber>
<DeptName>Alex</DeptName>
<usn>1RITCS001</usn>
<deptName>CSE</deptName>
</student>
<student>
<serialNumber>2</serialNumber>
<DeptName>Smith</DeptName>
<usn>1RITDS040</usn>
<deptName>DS</deptName>
</student>
<student>
<serialNumber>3</serialNumber>
<DeptName>Saliena</DeptName>
<usn>1RITCV051</usn>
<deptName>Civil</deptName>
</student>
<student>
<serialNumber>4</serialNumber>
<DeptName>Elizabeth</DeptName>
<usn>1RITME011</usn>
```

```
<deptName>Mechanical</deptName>
</student>
</students>
<departments>
<department>
<serialNumber>1</serialNumber>
<DeptName>CSE</DeptName>
<departmentId>01</departmentId>
<totalEnrolledStudents>170</totalEnrolledStudents>


</department>
<department>
<serialNumber>2</serialNumber>
<DeptName>DS</DeptName>
<departmentId>02</departmentId>


<totalEnrolledStudents>188</totalEnrolledStudents>
</department>
<department>
<serialNumber>3</serialNumber>

<DeptName>Civil</DeptName>
<departmentId>03</departmentId>
<totalEnrolledStudents>160</totalEnrolledStudents>
</department>
<department>
<serialNumber>4</serialNumber>
<DeptName>Mechanical</DeptName>
<departmentId>04</departmentId>
<totalEnrolledStudents>150</totalEnrolledStudents>
</department>
</departments>

</university>
```

```
exam@ThinkCentre-M70t:~$ cat sample.xml

exam@ThinkCentre-M70t:~$ xmlstarlet  sel -t -c "//student" sample.xml >
students.xml

Command 'xmlstarlet' not found, but can be installed with:

exam@ThinkCentre-M70t:~$ sudo snap install xmlstarlet # version 1.6.1+pkg- 47d2,
or

exam@ThinkCentre-M70t:~$ sudo apt install xmlstarlet # version 1.6.1-2build1


exam@ThinkCentre-M70t:~$ sudo apt install xmlstarlet # version 1.6.1-2build1




exam@ThinkCentre-M70t:~$ xmlstarlet sel -t -c "//student" sample.xml >
students.xml exam@ThinkCentre-M70t:~$ sed -i 's/<name>/<DeptName>/g'
sample.xml exam@ThinkCentre-M70t:~$ sed -i 's/<\/name>/<\/DeptName>/g'
sample.xml
exam@ThinkCentre-M70t:~$ nano replace_tags.sh
```

Copy and Paste the below code:

```
#!/bin/bash
sed -i 's/<name>/<DeptName>/g' sample.xml sed -i 's/<\/name>/<\/DeptName>/g'
sample.xml
```

Press Ctrl + X

```
exam@ThinkCentre-M70t:~$ chmod +x replace_tags.sh exam@ThinkCentre-
M70t:~$ ls replace_tags.sh exam@ThinkCentre-M70t:~$ ls
```

```
Desktop eclipse-workspace Public sample.xml Templates Documents Music
replace_tags.sh snap test
Downloads Pictures sample students.xml Videos

exam@ThinkCentre-M70t:~$ chmod +x replace_tags.sh exam@ThinkCentre-
M70t:~$ ./replace_tags.sh
exam@ThinkCentre-M70t:~$ xmlstarlet sel -t -v "//usn" sample.xml > usn_values.txt
exam@ThinkCentre-M70t:~$ cat students.xml
<student>
<serialNumber>1</serialNumber>
<name>Alex</name>
<usn>1RITCS001</usn>
<deptName>CSE</deptName>
</student><student>
<serialNumber>2</serialNumber>
<name>Smith</name>
<usn>1RITDS040</usn>

<deptName>DS</deptName>
</student><student>
<serialNumber>3</serialNumber>

<name>Saliena</name>


<usn>1RITCV051</usn>

<deptName>Civil</deptName>
</student><student>
<serialNumber>4</serialNumber>
<name>Elizabeth</name>
<usn>1RITME011</usn>
<deptName>Mechanical</deptName>
</student>exam@ThinkCentre-M70t:~$ cat usn_values.txt
1RITCS001

1RITDS040
```

12. Docker

a. Docker installation and set up
b. Create a shell script to pass arguments and run in a container using Docker.
c. Creating docker file
d. Building docker image, creating a container and running the shell scripts.
e. Displaying and running docker image.

Docker installation and set up administrator

```
@admin:~$ sudo apt-get update [sudo] password for administrator: Hit:1
http://security.ubuntu.com/ubuntu jammy-security InRelease Hit:2
http://in.archive.ubuntu.com/ubuntu jammy InRelease
Hit:3 http://in.archive.ubuntu.com/ubuntu jammy-updates InRelease Hit:4
http://in.archive.ubuntu.com/ubuntu jammy-backports InRelease Reading
package lists... Done

administrator@admin:~$ sudo apt-get install docker.io Reading package lists... Done
Building dependency tree... Done Reading state information... Done
docker.io is already the newest version (24.0.5-0ubuntu1~22.04.1).
0 upgraded, 0 newly installed, 0 to remove and 201 not upgraded.
administrator@admin:~$ sudo systemctl enable docker
 administrator@admin:~$ sudo systemctl status docker
        docker.service - Docker Application Container Engine
Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset:
enabled)
Active: active (running) since Mon 2024-02-19 10:39:52 IST; 2min 21s ago
TriggeredBy:

● docker.socket Docs: https://docs.docker.com
Main PID: 1663 (dockerd) Tasks: 29
Memory: 99.1M CPU: 596ms
CGroup: /system.slice/docker.service
 └─1663 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
Feb 19 10:39:51 admin dockerd[1663]: time="2024-02-
19T10:39:51.918786750+05:30" level=info msg="Starting up"
Feb 19 10:39:51 admin dockerd[1663]: time="2024-02-
19T10:39:51.919957523+05:30" level=info msg="detected 127.0.0.53
nameserver, assuming systemd-resolved, so using resolv.conf:
/run/systemd/resolve/reso>
Feb 19 10:39:51 admin dockerd[1663]: time="2024-02-
```

```
19T10:39:51.950714941+05:30" level=info msg="[graphdriver] using prior
storage driver: overlay2"
Feb 19 10:39:51 admin dockerd[1663]: time="2024-02-
19T10:39:51.951294483+05:30" level=info msg="Loading containers: start."
Feb 19 10:39:52 admin dockerd[1663]: time="2024-02-
19T10:39:52.518925979+05:30" level=info msg="Default bridge (docker0) is
assigned with an IP address 172.17.0.0/16. Daemon option --bip can be used to >
Feb 19 10:39:52 admin dockerd[1663]: time="2024-02-
19T10:39:52.597010094+05:30" level=info msg="Loading containers: done."
Feb 19 10:39:52 admin dockerd[1663]: time="2024-02-
19T10:39:52.622350224+05:30" level=info msg="Docker daemon"
commit="24.0.5-0ubuntu1~22.04.1" graphdriver=overlay2 version=24.0.5 Feb 19
10:39:52 admin dockerd[1663]: time="2024-02-19T10:39:52.622639406+05:30"
level=info
msg="Daemon has completed initialization"
Feb 19 10:39:52 admin dockerd[1663]: time="2024-02-
19T10:39:52.660608287+05:30" level=info msg="API listen on /run/docker.sock"
Feb 19 10:39:52 admin systemd[1]: Started Docker Application Container Engine.
lines 1-22/22 (END)
[1]+ Stopped sudo systemctl status docker administrator@admin:~$ sudo
service docker start administrator@admin:~$ sudo systemctl start docker
administrator@admin:~$ sudo docker run hello-world

Unable to find image 'hello-world:latest' locally latest: Pulling from library/hello-
world c1ec31eb5944: Pull complete

Digest:
sha256:d000bc569937abbe195e20322a0bde6b2922d805332fd6d8a68b19f524b7d
21d Status: Downloaded newer image for hello-world:latest
Hello from Docker!
This message shows that your installation appears to be working correctly.
To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker
Hub. (amd64)
3. The Docker daemon created a new container from that image which runs
the executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which
sent it to your terminal.
To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash
Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/
For more examples and ideas, visit: https://docs.docker.com/get-started/
```

b. Create a shell script to pass arguments and run in a container using Docker.

```
administrator@admin:~$ vi myscript.sh
#!/bin/bash
echo "Hello, Docker! Argument passed: $1"
administrator@admin:~$ sh myscript.sh Hello, Docker! Argument passed:
```

c.Creating docker file

A Dockerfile is a text document that contains instructions for building a Docker image. Here's an example Dockerfile for a simple Python application:

Create a textfile without any extension: vi Dockerfile

```
# Use an official Python runtime as a parent image FROM python:3.9-slim
```

# Set the working directory in the container WORKDIR /app
# Copy the current directory contents into the container at /app COPY . /app
# Install any needed dependencies specified in requirements.txt RUN pip install --no-cache-dir -r requirements.txt
# Define environment variable ENV ENVIRONMENT=production
# Run app.py when the container launches CMD ["python", "app.py"]
Save this file as Dockerfile in your project directory.
d.Building docker image, creating a container and running the shell scripts.
Run Docker Commands as Root or with sudo: If you're running Docker commands as a regular user, prefix the commands with sudo to run them with root privileges. For example:
sudo docker build -t my_python_app .
Add Your User to the Docker Group (Linux only): You can add your user to the docker group, which allows you to execute Docker commands without using sudo. However, be cautious as this grants your user elevated permissions to Docker.

First, check if the docker group exists:
grep docker /etc/group
If the docker group does not exist, you'll need to create it:
sudo groupadd docker
Then, add your user to the docker group:
sudo usermod -aG docker $USER

e. Displaying and running docker image.
To display information about the Docker image you just built, you can use the following command:
docker images