



SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
GENERAL SEMESTER 2023-24

B.Tech - CSE

BCSE303P: Operating Systems Lab

1. a. Study of basic LINUX commands (refer Pdf for list of commands)
- b. Shell Programming (refer Pdf for list of experiments)
- c. Process System calls. {Orphan, Zombie, etc.}
- i. fork (), exit (), kill (), exec (), wait (), sleep () - Programming

Language: C or C++

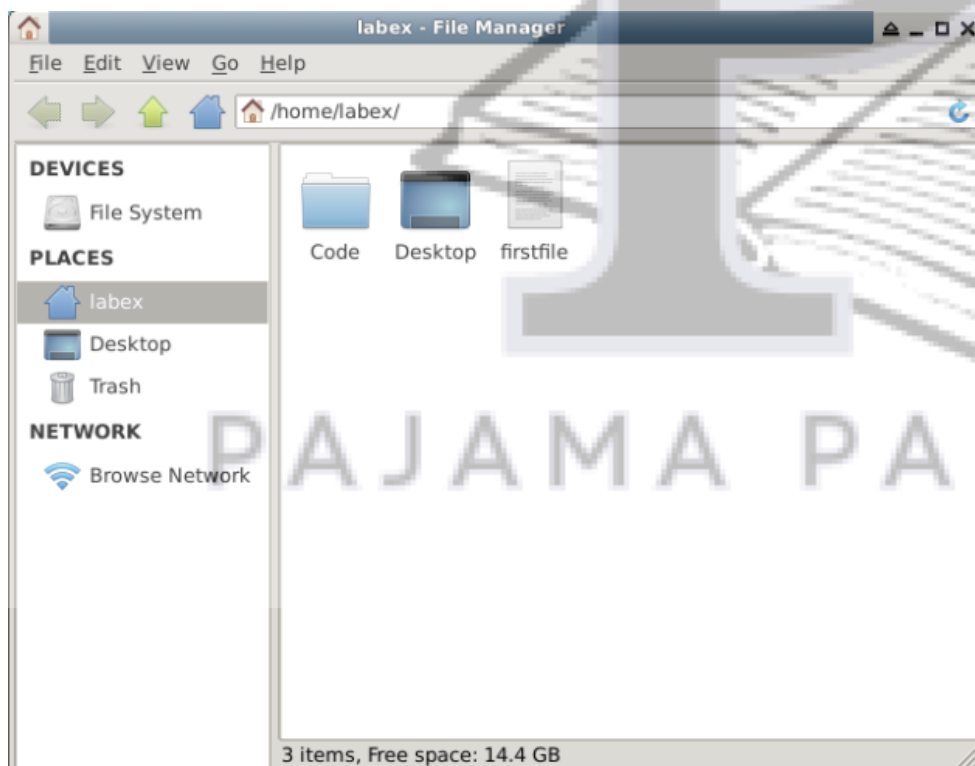
Study of Basic Linux Commands

Creating a File

cat > firstfile

```
Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 5.15.0-56-generic x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:        https://ubuntu.com/advantage
Last login: Sun Apr 30 02:06:22 2023 from 172.17.0.2
labex:~/ $ cat>firstfile [13:12:44]
```



Copying a File

cp firstfile secondfile

```
labex:~/ $ cat>firstfile [13:12:44]
This is just a test
labex:~/ $ [13:14:32]
labex:~/ $ cp firstfile secondfile [13:14:43]
labex:~/ $ [13:16:44]
```



Renaming a File

mv secondfile file2

```
Last login: Sun Apr 30 02:00:22 2023 from 172.17.0.2
labex:~/ $ cat>firstfile [13:12:44]
This is just a test
labex:~/ $ [13:14:32]
labex:~/ $ cp firstfile secondfile [13:14:43]
labex:~/ $ mv secondfile file2 [13:16:44]
labex:~/ $ [13:18:36]
```



Removing a File

rm file2

```
labex:~/ $ cat>firstfile [13:12:44]
This is just a test
labex:~/ $ [13:14:32]
labex:~/ $ cp firstfile secondfile [13:14:43]
labex:~/ $ mv secondfile file2 [13:16:44]
labex:~/ $ rm file2 [13:18:36]
labex:~/ $ [13:20:01]
```



Creating a Directory

mkdir project1

```
labex:~/ $ mkdir project1 [13:20:01]
labex:~/ $ [13:22:39]
```

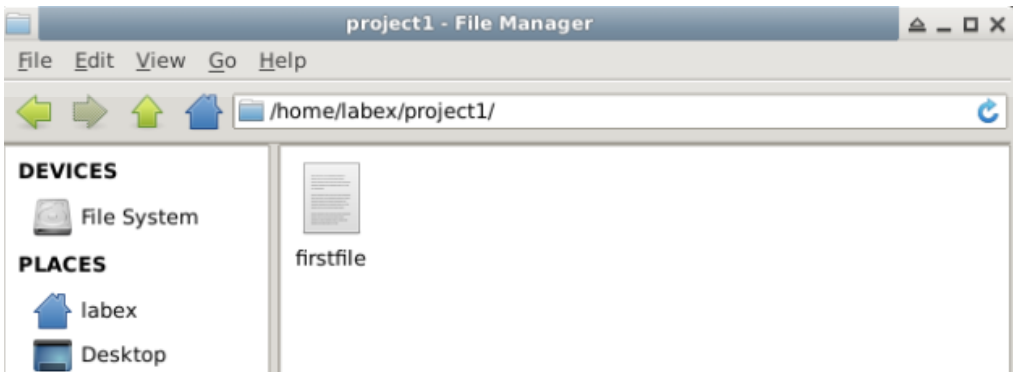


Moving and Copying Files Into a Directory

mv firstfile project1

cp firstfile project1

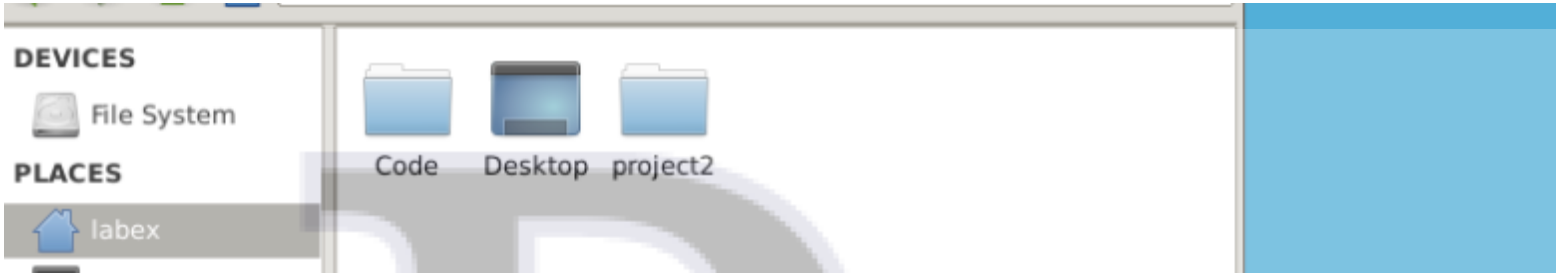
```
labex:~/ $ mv firstfile project1 [13:22:39]
labex:~/ $ [13:25:20]
```



Renaming a Directory

mv project1 project2

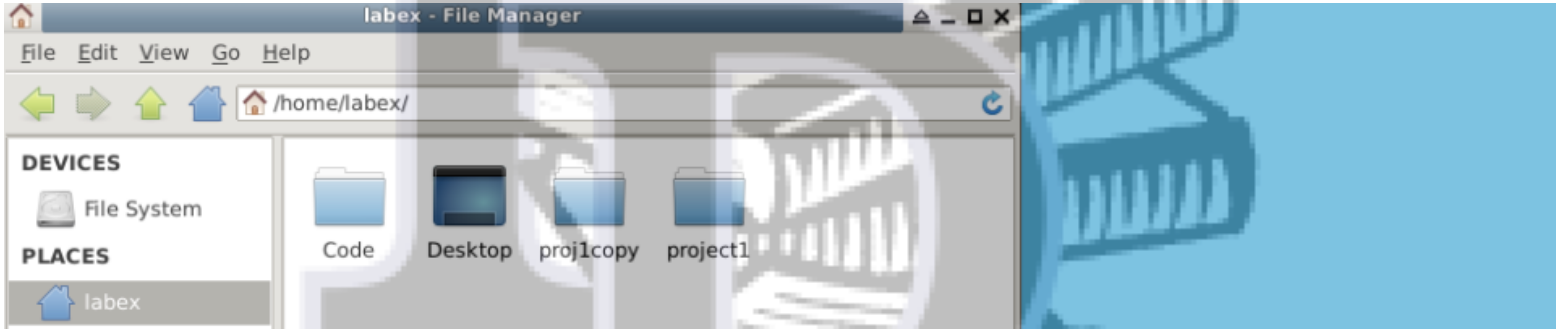
```
labex:~/ $ mv project1 project2 [13:25:20]
labex:~/ $ [13:26:32]
```



Copying a Directory

cp -r project1 proj1copy

```
labex:~/ $ cp -r project1 proj1copy [13:31:11]
labex:~/ $ [13:32:01]
```



Removing a Directory

rmdir testdir1 testdir2

If you try to remove a directory that is not empty, you will see

rmdir: testdir3: Directory not empty

```
labex:~/ $ rmdir: testdir3: [13:32:01]
zsh: command not found: rmdir:
labex:~/ $ [13:34:09]
```

Summary of Commands

Working With Files

- **mv file1 file2**
Renames **file1** to **file2** (if **file2** existed previously, overwrites original contents of **file2**).
- **cp file1 file2**
Copies **file1** as **file2** (if **file2** existed previously, overwrites original contents of **file2**).
- **rm file3 file4**
Removes **file3** and **file4**, requesting confirmation for each removal.

Working With Directories

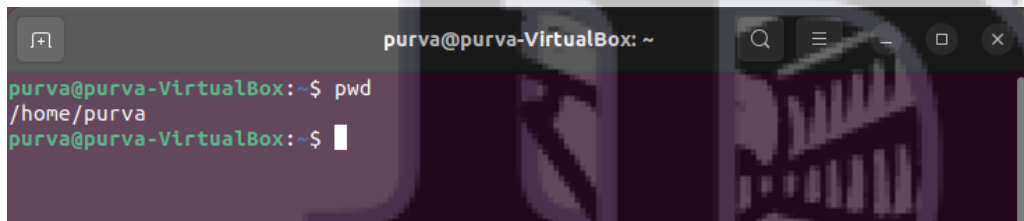
- **mkdir dir1**
Creates a new directory called **dir1**.
- **mv dir1 dir2**
If **dir2** does not exist, renames **dir1** to **dir2**.
If **dir2** does exist, moves **dir1** inside **dir2**.
- **cp -r dir1 dir2**
If **dir2** does not exist, copies **dir1** as **dir2**.
If **dir2** does exist, copies **dir1** inside **dir2**.
- **rmdir dir1**
Removes **dir1**, if **dir1** contains no files.
- **rm -r dir1**
Removes **dir1** and any files it contains. Use with caution.

Working With Files and Directories

- **cp file1 dir1**
Copies file **file1** into existing directory **dir1**.
- **mv file2 dir2**
Moves file **file2** into existing directory **dir2**.

Determining Which Directory You're Currently In

pwd

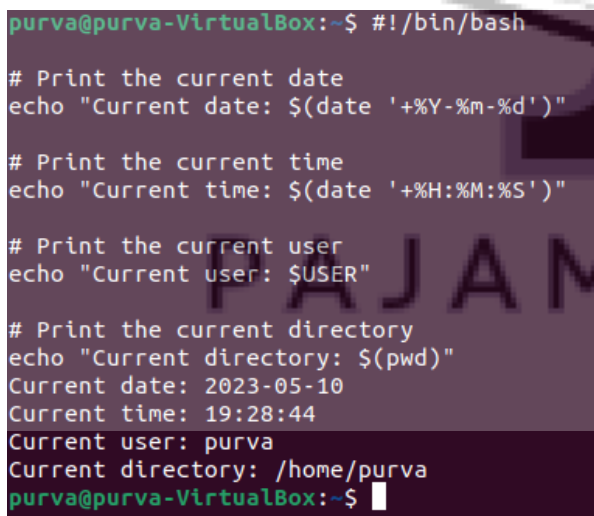


```
purva@purva-VirtualBox: ~  
purva@purva-VirtualBox:~$ pwd  
/home/purva  
purva@purva-VirtualBox:~$
```

SHELL PROGRAMMING

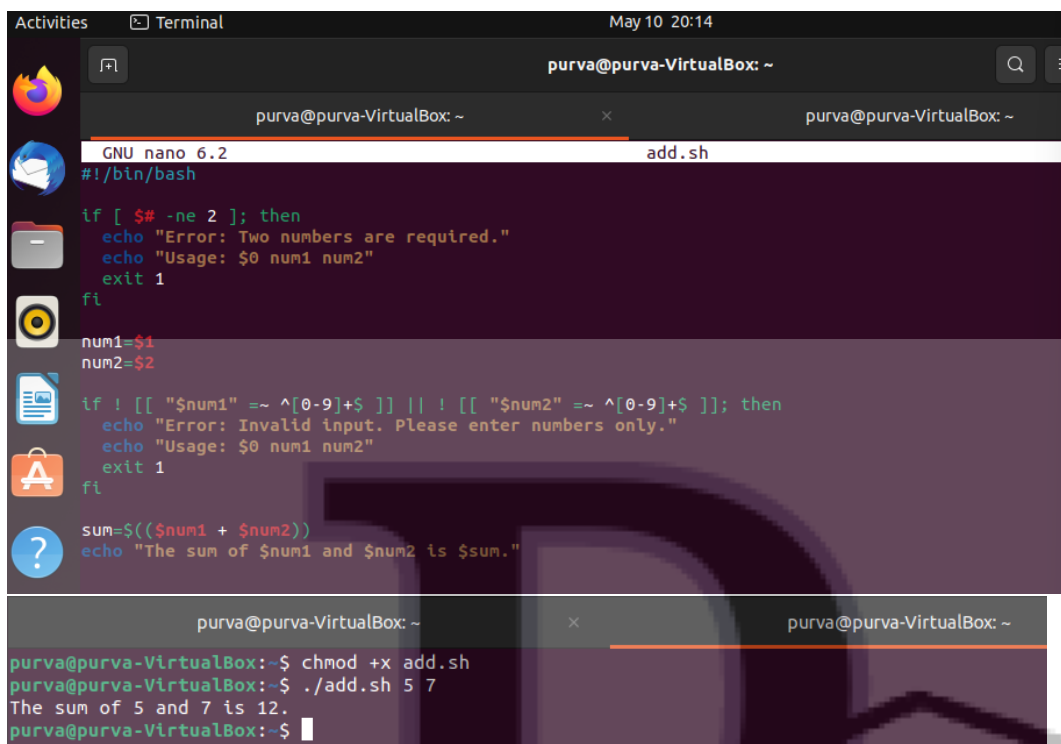
Exercise problems

1. Write Script to see current date, time, username, and current directory



```
purva@purva-VirtualBox:~$ #!/bin/bash  
# Print the current date  
echo "Current date: $(date '+%Y-%m-%d')"  
# Print the current time  
echo "Current time: $(date '+%H:%M:%S')"  
# Print the current user  
echo "Current user: $USER"  
# Print the current directory  
echo "Current directory: $(pwd)"  
Current date: 2023-05-10  
Current time: 19:28:44  
Current user: purva  
Current directory: /home/purva  
purva@purva-VirtualBox:~$
```

2. How to write shell script that will add two numbers, which are supplied as command line argument, and if this two numbers are not given show error and its usage.



The screenshot shows a terminal window with two tabs. The first tab, titled 'add.sh', contains the following code in nano 6.2:

```
#!/bin/bash

if [ $# -ne 2 ]; then
    echo "Error: Two numbers are required."
    echo "Usage: $0 num1 num2"
    exit 1
fi

num1=$1
num2=$2

if ! [[ "$num1" =~ ^[0-9]+$ ]] || ! [[ "$num2" =~ ^[0-9]+$ ]]; then
    echo "Error: Invalid input. Please enter numbers only."
    echo "Usage: $0 num1 num2"
    exit 1
fi

sum=$((num1 + num2))
echo "The sum of $num1 and $num2 is $sum."
```

The second tab shows the execution of the script:

```
purva@purva-VirtualBox:~$ chmod +x add.sh
purva@purva-VirtualBox:~$ ./add.sh 5 7
The sum of 5 and 7 is 12.
purva@purva-VirtualBox:~$
```

3. Write Script to find out biggest number from given three nos. Numbers are supplied as commandline argument. Print error if sufficient arguments are not supplied.



The screenshot shows a terminal window with two tabs. The first tab, titled 'big.sh', contains the following code in nano 6.2:

```
#!/bin/bash

if [ $# -ne 3 ]; then
    echo "Error: Three numbers are required."
    echo "Usage: $0 num1 num2 num3"
    exit 1
fi

num1=$1
num2=$2
num3=$3

if ! [[ "$num1" =~ ^[0-9]+$ ]] || ! [[ "$num2" =~ ^[0-9]+$ ]] || ! [[ "$num3" =~ ^[0-9]+$ ]]; then
    echo "Error: Invalid input. Please enter numbers only."
    echo "Usage: $0 num1 num2 num3"
    exit 1
fi

if [ $num1 -gt $num2 ] && [ $num1 -gt $num3 ]; then
    echo "$num1 is the biggest number."
elif [ $num2 -gt $num1 ] && [ $num2 -gt $num3 ]; then
    echo "$num2 is the biggest number."
else
    echo "$num3 is the biggest number."
fi
```

The second tab shows the execution of the script:

```
purva@purva-VirtualBox:~$ chmod +x big.sh
purva@purva-VirtualBox:~$ ./big.sh 5 7 3
7 is the biggest number.
purva@purva-VirtualBox:~$
```

4. Write script to print the following numbers as 5,4,3,2,1 using while loop.


```
purva@purva-VirtualBox: ~  
GNU nano 6.2 reverse.sh  
#!/bin/bash  
  
echo "Enter a number:"  
read number  
  
reverse=""  
  
while [ $number -gt 0 ]  
do  
    remainder=$((number % 10))  
    reverse="$reverse$remainder"  
    number=$((number / 10))  
done  
  
echo "Reverse of the number is $reverse"
```

```
purva@purva-VirtualBox: ~  
purva@purva-VirtualBox:~$ chmod +x reverse.sh  
purva@purva-VirtualBox:~$ ./reverse.sh  
Enter a number:  
543  
Reverse of the number is 345  
purva@purva-VirtualBox:~$ ./reverse.sh  
Enter a number:  
987  
Reverse of the number is 789  
purva@purva-VirtualBox:~$
```

7. Write script to print given numbers sum of all digit, for eg. If no is 123 it's sum of all digit will be $1+2+3 = 6$.

```
purva@purva-VirtualBox: ~  
GNU nano 6.2 sum.sh  
#!/bin/bash  
  
echo "Enter a number:"  
read number  
  
sum=0  
  
while [ $number -gt 0 ]  
do  
    remainder=$((number % 10))  
    sum=$((sum + remainder))  
    number=$((number / 10))  
done  
  
echo "Sum of all digits in the number is $sum"
```

```
purva@purva-VirtualBox: ~  
purva@purva-VirtualBox:~$ chmod +x sum.sh  
purva@purva-VirtualBox:~$ ./sum.sh  
Enter a number:  
564  
Sum of all digits in the number is 15  
purva@purva-VirtualBox:~$ ./sum.sh  
Enter a number:  
21  
Sum of all digits in the number is 3  
purva@purva-VirtualBox:~$
```

8. Write script to determine whether given file exist or not, file name is supplied as command line argument, also check for sufficient number of command line argument.

PAJAMA PADHAI


```
purva@purva-VirtualBox: ~
GNU nano 6.2 checkfile.sh
#!/bin/bash

if [ $# -eq 0 ]; then
    echo "Error: No file name provided."
    echo "Usage: ./checkfile.sh <file_name>"
    exit 1
fi

filename=$1

if [ -f "$filename" ]; then
    echo "File $filename exists."
else
    echo "File $filename does not exist."
fi
```

```
purva@purva-VirtualBox: ~
purva@purva-VirtualBox:~$ chmod +x checkfile.sh
purva@purva-VirtualBox:~$ ./checkfile.sh test.txt
File test.txt does not exist.
purva@purva-VirtualBox:~$
```

9. Write a shell script takes the name a path (eg: /afs/andrew/course/15/123/handin), and counts all the sub directories (recursively).

```
purva@purva-VirtualBox: ~
GNU nano 6.2 countdirs.sh
#!/bin/bash

if [ $# -eq 0 ]; then
    echo "Error: No path provided."
    echo "Usage: ./countdirs.sh <path>"
    exit 1
fi

path=$1
count=0

if [ -d "$path" ]; then
    count=$(find "$path" -type d | wc -l)
    count=$((count - 1)) # Subtract 1 for the top-level directory
    echo "Number of subdirectories in $path: $count"
else
    echo "Error: $path is not a valid directory."
    exit 1
fi
```

```
purva@purva-VirtualBox: ~
purva@purva-VirtualBox:~$ chmod +x countdirs.sh
purva@purva-VirtualBox:~$ ./countdirs.sh /afs/andrew/course/15/123/handin
Error: /afs/andrew/course/15/123/handin is not a valid directory.
purva@purva-VirtualBox:~$
```

10. Write a shell script that takes a name of a folder as a command line argument, and produce a file that contains the names of all sub folders with size 0 (that is empty sub folders)

```
purva@purva-VirtualBox: ~
GNU nano 6.2 emptyfolders.sh
#!/bin/bash

if [ $# -eq 0 ]; then
    echo "Error: No folder name provided."
    echo "Usage: ./emptyfolders.sh <folder name>"
    exit 1
fi

if [ ! -d "$1" ]; then
    echo "Error: $1 is not a valid directory."
    exit 1
fi

find "$1" -type d -empty > empty_folders.txt
echo "Empty folders list saved to empty_folders.txt"
```

```
purva@purva-VirtualBox: ~
purva@purva-VirtualBox:~$ chmod +x emptyfolders.sh
purva@purva-VirtualBox:~$ ./emptyfolders.sh /path/to/folder
Error: /path/to/folder is not a valid directory.
purva@purva-VirtualBox:~$
```

11. Write a shell script that takes a name of a folder, and delete all sub folders of size 0.

```
purva@purva-VirtualBox: ~  
GNU nano 6.2 delete_empty_folders.sh  
#!/bin/bash  
  
if [ $# -eq 0 ]; then  
    echo "Error: No folder name provided."  
    echo "Usage: ./delete_empty_folders.sh <folder name>"  
    exit 1  
fi  
  
if [ ! -d "$1" ]; then  
    echo "Error: $1 is not a valid directory."  
    exit 1  
fi  
  
find "$1" -type d -empty -delete  
echo "Empty folders deleted."
```

```
purva@purva-VirtualBox: ~  
purva@purva-VirtualBox:~$ chmod +x delete_empty_folders.sh  
purva@purva-VirtualBox:~$ ./delete_empty_folders.sh /path/to/folder  
Error: /path/to/folder is not a valid directory.  
purva@purva-VirtualBox:~$
```

12. Write a shell script that will take an input file and remove identical lines (or duplicate lines from the file)

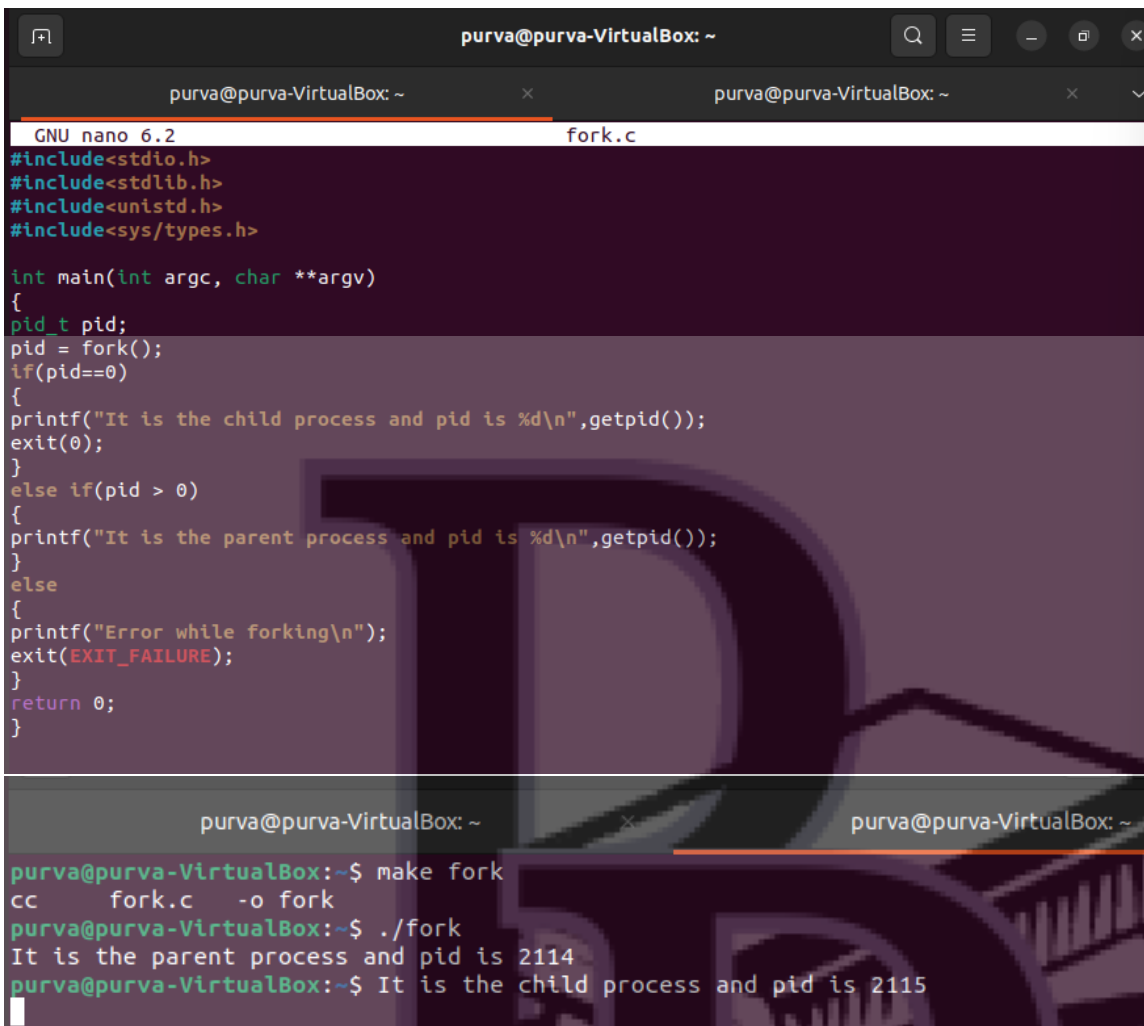
```
purva@purva-VirtualBox: ~  
GNU nano 6.2 remove_duplicates.sh  
#!/bin/bash  
  
if [ $# -eq 0 ]; then  
    echo "Error: No filename provided."  
    echo "Usage: ./remove_duplicates.sh <filename>"  
    exit 1  
fi  
  
if [ ! -f "$1" ]; then  
    echo "Error: $1 is not a valid file."  
    exit 1  
fi  
  
sort -u "$1" -o "$1"  
echo "Duplicate lines removed."
```

```
purva@purva-VirtualBox: ~  
purva@purva-VirtualBox:~$ chmod +x remove_duplicates.sh  
purva@purva-VirtualBox:~$ ./remove_duplicates.sh input.txt  
Error: input.txt is not a valid file.  
purva@purva-VirtualBox:~$
```

Process System calls. {Orphan, Zombie, etc.}

fork ()

PAJAMA PADHAI



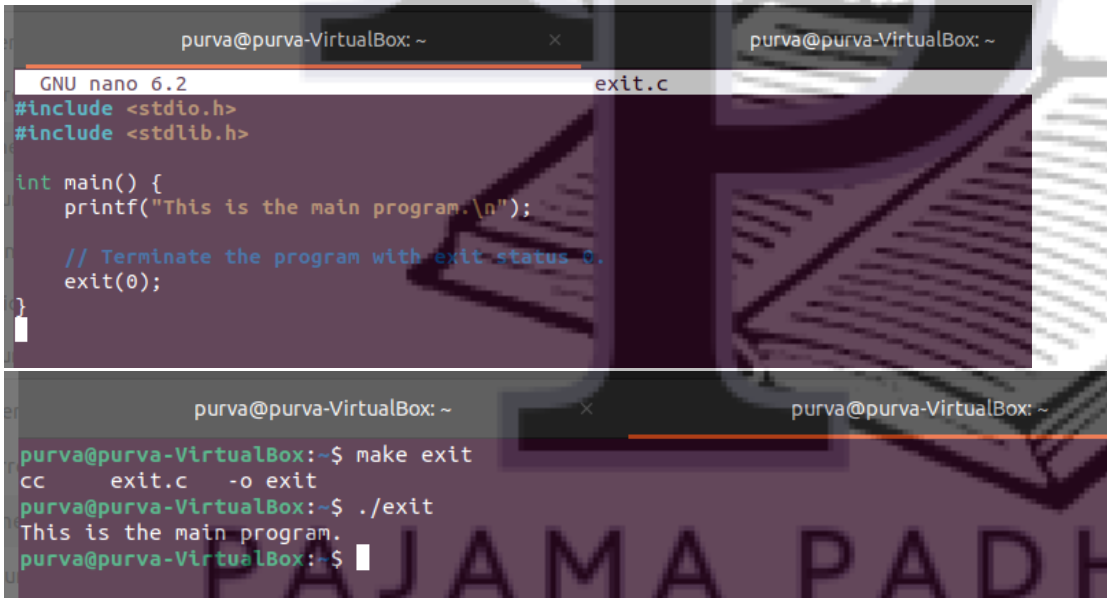
The screenshot shows a terminal window with two tabs. The first tab, titled 'GNU nano 6.2 fork.c', displays the source code for a C program that demonstrates the `fork()` system call. The code includes headers for `stdio.h`, `stdlib.h`, `unistd.h`, and `sys/types.h`. The `main` function takes `argc` and `argv` as arguments. It declares a `pid_t` variable, calls `fork()` to create a child process, and then uses `getpid()` to print the process ID. The parent process prints 'It is the parent process and pid is 2114' and the child process prints 'It is the child process and pid is 2115'. The second tab is empty. Below the terminal window, the command `make fork` is executed, followed by `./fork`, which produces the expected output.

```
GNU nano 6.2 fork.c
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/types.h>

int main(int argc, char **argv)
{
    pid_t pid;
    pid = fork();
    if(pid==0)
    {
        printf("It is the child process and pid is %d\n",getpid());
        exit(0);
    }
    else if(pid > 0)
    {
        printf("It is the parent process and pid is %d\n",getpid());
    }
    else
    {
        printf("Error while forking\n");
        exit(EXIT_FAILURE);
    }
    return 0;
}
```

```
purva@purva-VirtualBox:~$ make fork
cc fork.c -o fork
purva@purva-VirtualBox:~$ ./fork
It is the parent process and pid is 2114
purva@purva-VirtualBox:~$ It is the child process and pid is 2115
```

exit ()



The screenshot shows a terminal window with two tabs. The first tab, titled 'GNU nano 6.2 exit.c', displays the source code for a C program that demonstrates the `exit()` system call. The code includes headers for `stdio.h` and `stdlib.h`. The `main` function prints 'This is the main program.\n' and then calls `exit(0)` to terminate the program. The second tab is empty. Below the terminal window, the command `make exit` is executed, followed by `./exit`, which produces the expected output.

```
GNU nano 6.2 exit.c
#include <stdio.h>
#include <stdlib.h>

int main() {
    printf("This is the main program.\n");

    // Terminate the program with exit status 0.
    exit(0);
}
```

```
purva@purva-VirtualBox:~$ make exit
cc exit.c -o exit
purva@purva-VirtualBox:~$ ./exit
This is the main program.
purva@purva-VirtualBox:~$
```

kill ()

purva@purva-VirtualBox: ~

purva@purva-VirtualBox: ~

GNU nano 6.2

kill.c

```
#include <stdio.h>
#include <signal.h>
#include <unistd.h>

int main() {
    pid_t pid;
    int sig;

    printf("Enter PID of process to send signal to: ");
    scanf("%d", &pid);

    printf("Enter signal to send (e.g. 9 for SIGKILL): ");
    scanf("%d", &sig);

    if (kill(pid, sig) == 0) {
        printf("Signal sent successfully.\n");
    } else {
        printf("Failed to send signal.\n");
    }

    return 0;
}
```

purva@purva-VirtualBox: ~

purva@purva-VirtualBox: ~

```
purva@purva-VirtualBox:~$ make kill
cc    kill.c  -o kill
purva@purva-VirtualBox:~$ ./kill
Enter PID of process to send signal to: 12
Enter signal to send (e.g. 9 for SIGKILL): 5
Failed to send signal.
purva@purva-VirtualBox:~$
```

exec ()

PAJAMA PADHAI

purva@purva-VirtualBox: ~

purva@purva-Virtual

GNU nano 6.2

exec.c

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>

main(void) {
    pid_t pid = 0;
    int status;
    pid = fork();

    if (pid == 0) {
        printf("I am the child.");
        execl("/bin/ls", "ls", "-l", "/home/ubuntu/", (char *) 0);
        perror("In exec(): ");
    }

    if (pid > 0) {
        printf("I am the parent, and the child is %d.\n", pid);
        pid = wait(&status);
        printf("End of process %d: ", pid);

        if (WIFEXITED(status)) {
            printf("The process ended with exit(%d).\n", WEXITSTATUS(status));
        }

        if (WIFSIGNALED(status)) {
            printf("The process ended with kill -%d.\n", WTERMSIG(status));
        }
    }

    if (pid < 0) {
        perror("In fork():");
    }

    exit(0);
}
```

purva@purva-VirtualBox: ~

purva@purva-VirtualBox: ~

```
purva@purva-VirtualBox:~$ make exec
cc      exec.c      -o exec
exec.c:7:1: warning: return type defaults to 'int' [-Wimplicit-int]
   7 | main(void) {
     | ~~~~
purva@purva-VirtualBox:~$ ./exec
I am the parent, and the child is 2540.
ls: cannot access '/home/ubuntu/': No such file or directory
End of process 2540: The process ended with exit(2).
purva@purva-VirtualBox:~$
```

wait ()

PAJAMA PADHAI

```
GNU nano 6.2 wait.c
#include<stdio.h> // printf()
#include<stdlib.h> // exit()
#include<sys/types.h> // pid_t
#include<sys/wait.h> // wait()
#include<unistd.h> // fork

int main(int argc, char **argv)
{
    pid_t pid;
    pid = fork();

    if(pid==0)
    {
        printf("It is the child process and pid is %d\n",getpid());

        int i=0;
        for(i=0;i<8;i++)
        {
            printf("%d\n",i);
        }

        exit(0);
    }
    else if(pid > 0)
    {
        printf("It is the parent process and pid is %d\n",getpid());

        int status;
        wait(&status);
        printf("Child is reaped\n");
    }
    else
    {
    }
}
```

```
purva@purva-VirtualBox: ~
purva@purva-VirtualBox:~$ make wait
cc wait.c -o wait
purva@purva-VirtualBox:~$ ./wait
It is the parent process and pid is 2576
It is the child process and pid is 2577
0
1
2
3
4
5
6
7
Child is reaped
purva@purva-VirtualBox:~$
```

sleep ()

```
GNU nano 6.2 sleep.c
#include <stdio.h>
#include <unistd.h>

int main() {
    printf("Starting the program...\n");
    sleep(5);
    printf("Woke up after sleeping for 5 seconds.\n");

    return 0;
}
```

```
purva@purva-VirtualBox:~$ make sleep
cc sleep.c -o sleep
purva@purva-VirtualBox:~$ ./sleep
Starting the program...
Woke up after sleeping for 5 seconds.
purva@purva-VirtualBox:~$
```




PAJAMA PADHAI