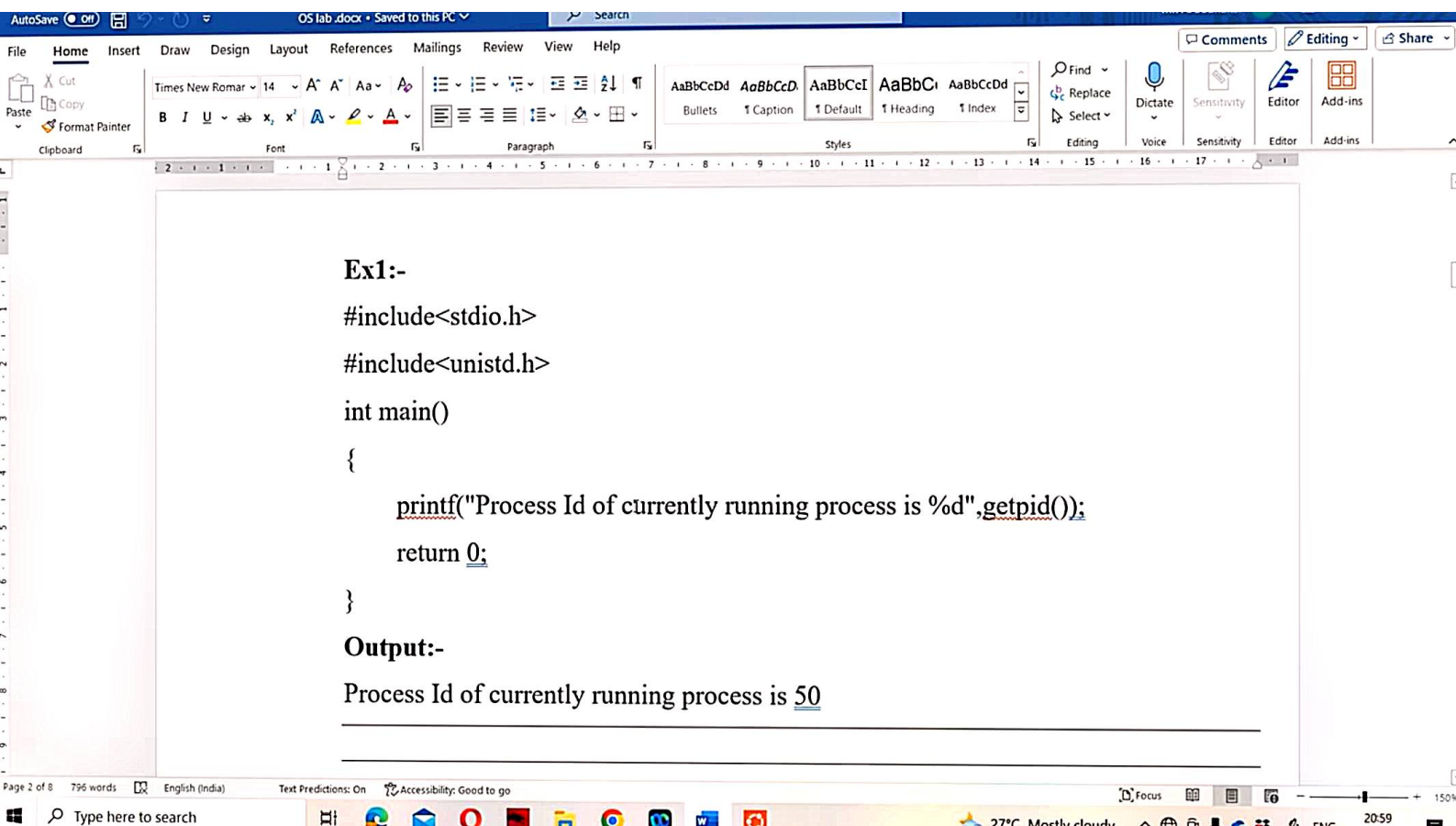


### 1.fork() System Call:-

- It is used to create a new process, known as the child process, which is an exact replica or duplicate or copy of the parent process.
- The process from which where we call fork() system call is known as parent process.
- The duplicate process that is created is called as child process.
- Parent process and child process will have different Process ID's.
- After the fork() system call, both the parent and the child processes run independently, each with its own memory space and resources.
- The fork() system call is declared in the <unistd.h> header file.
- The fork() system call has the following return values:
  - **On success**, it returns twice:

- The duplicate process that is created is called as child process.
- Parent process and child process will have different Process ID's.
- After the fork() system call, both the parent and the child processes run independently, each with its own memory space and resources.
- The fork() system call is declared in the <unistd.h> header file.
- The fork() system call has the following return values:
  - **On success**, it returns twice:
    - To the **parent process**, it returns the process ID (PID) of the newly created child process. This PID is a positive integer and is used by the parent process to refer to the child process in future system calls.
    - To the **child process**, it returns 0. This is how the child process can know that it is the child.
  - **On failure**, it returns -1 to the parent process, no child process is created.



Process Id of currently running process is 20

---

**Ex2:-**

```
#include<stdio.h>
#include<unistd.h>
int main()
{
    fork();
    printf("Process Id is %d", getpid());
    return 0;
}
```

**Output:-**

Process Id is 450

Page 2 of 8 796 words English (India) Text Predictions: On Accessibility: Good to go

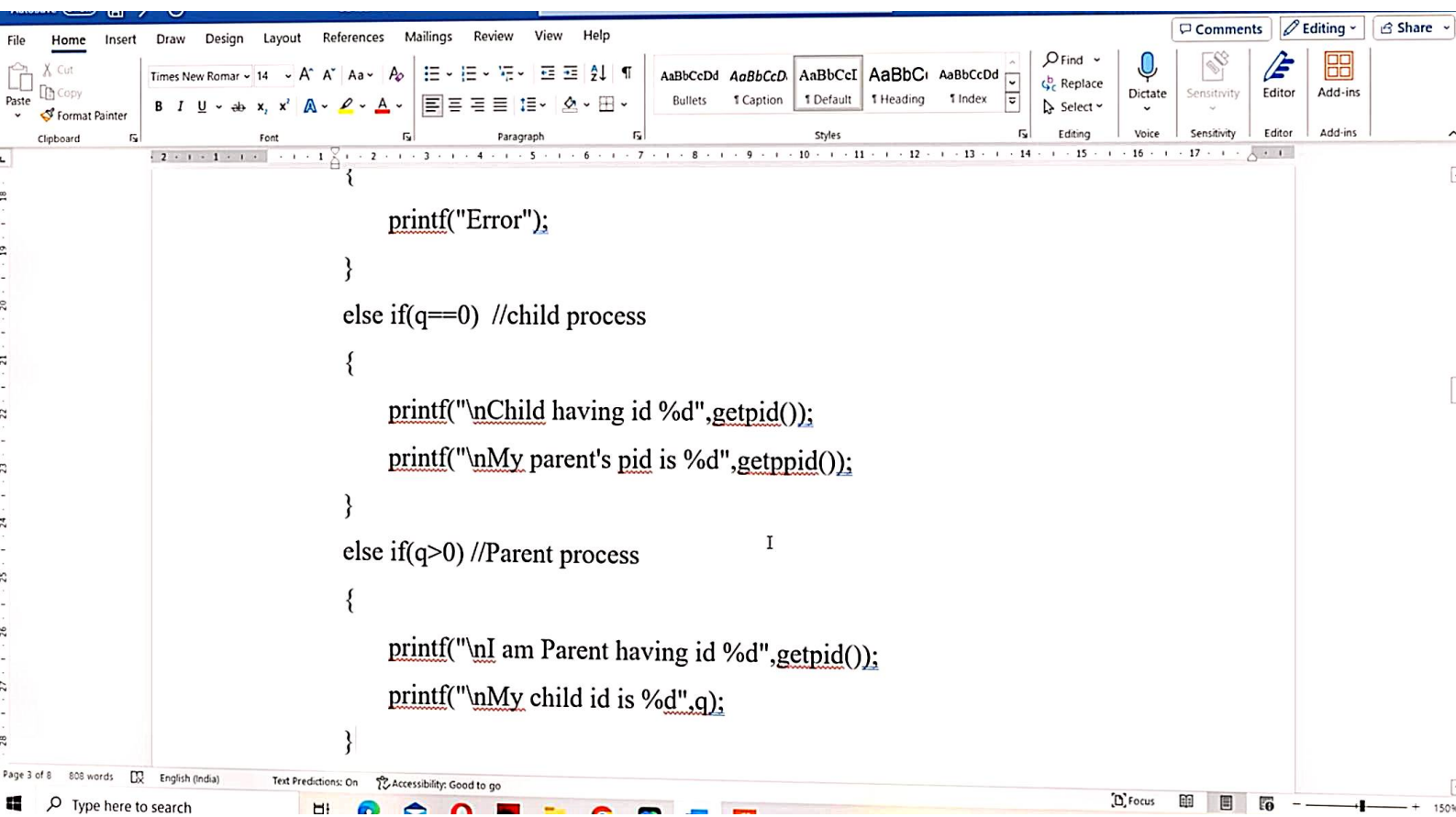
Ex3:-

```
#include<stdio.h>
#include<unistd.h>
int main()
{
    fork(); p1 p2
    fork(); p3 p4
    fork(); 1-5, 2-6 3-7 4-8
    printf("\nhello my process id is %d",getpid());
    return 0;
```

```
printf("Error");
```

Page 3 of 8 808 words English (India) Text Predictions: On Accessibility: Good to go







**Output:-**

Autosave 05:10:200x Search

File Home Insert Draw Design Layout References Mailings Review View Help

Comments Editing Share

Clipboard Font Paragraph Styles Editing Voice Sensitivity Editor Add-ins

Times New Roman 14 A<sup>+</sup> A<sup>-</sup> Aa Font

B I U x<sub>2</sub> x<sup>2</sup> Paragraph

AaBbCcDd AaBbCcD AaBbCcI AaBbCcI AaBbCcDd Styles

Bullets 1 Caption 1 Default 1 Heading 1 Index

Find Replace Select Dictate Voice Sensitivity Editor Add-ins

printf("\nCommon");

}

**Output:-**

I am Parent having id 4402

My child is 4403

Common

Child having id 4403

My parent's pid is 4402

Common

Page 4 of 8 1 of 608 words English (India) Text Predictions: On Accessibility: Good to go

Type here to search

Focus 150%

```
sudhakaratchala@MSI:~$ ./a.out
```

```
hello my process id is 50
```

```
hello my process id is 54hello my process id is 55hello my process id is 53hello my p  
rocess id is 51hello my process id is 57hello my process id is 52hello my process id i  
s 56sudhakaratchala@MSI:~$ pico ex4.c  
sudhakaratchala@MSI:~$ gcc ex4.c  
sudhakaratchala@MSI:~$ ./a.out
```

```
I am Parent having id 64
```

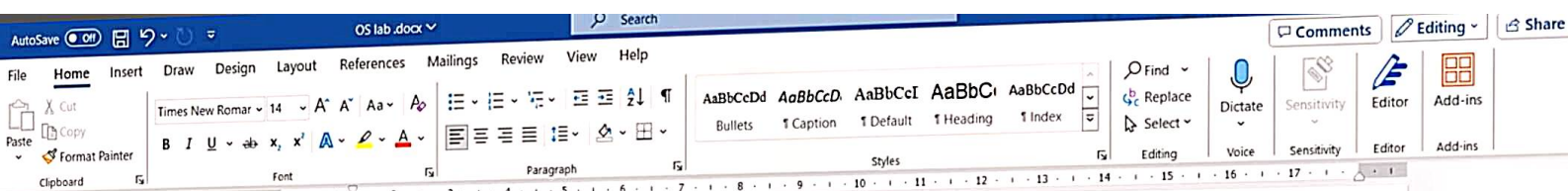
```
Child having id 65
```

```
My child id is 65
```

```
My parent's pid is 64
```

```
common common sudhakaratchala@MSI:~$
```

Windows taskbar with search bar and task icons.



## **2.wait() System call:-**

The wait() system call suspends execution of the calling process i.e, parent process until one of its children completed its execution.

```
#include<stdio.h>      I
#include<unistd.h>
#include<sys/types.h>
#include<sys/wait.h>
int main()
{
    pid_t q;
    q=fork();
```

```
if(q<0)
{
    printf("Error");
}
else if(q==0) //child process
{
    printf("\nChild having id %d",getpid());
    printf("\nMy parent's pid is %d",getppid());
}
else if(q>0) //Parent process
{
    wait(NULL);
    printf("\nI am Parent having id %d",getpid());
    printf("\nMy child is %d",q);
}
printf("\ncommon");
return 0;
```

<b>^G</b> Help	<b>^O</b> Write Out	<b>^W</b> Where Is	<b>^K</b> Cut	<b>^T</b> Execute	<b>^C</b> Location
<b>^X</b> Exit	<b>^R</b> Read File	<b>^_\</b> Replace	<b>^U</b> Paste	<b>^J</b> Justify	<b>^/</b> Go To Line

### 3.exec() System Call:-

The exec family of system calls is used to replace the current process image with a new process image.

Ex:- execv(), execlp()

#### execv() System Call:-

syntax:- int execv(const char \*path, char \*const argv[]);

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

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

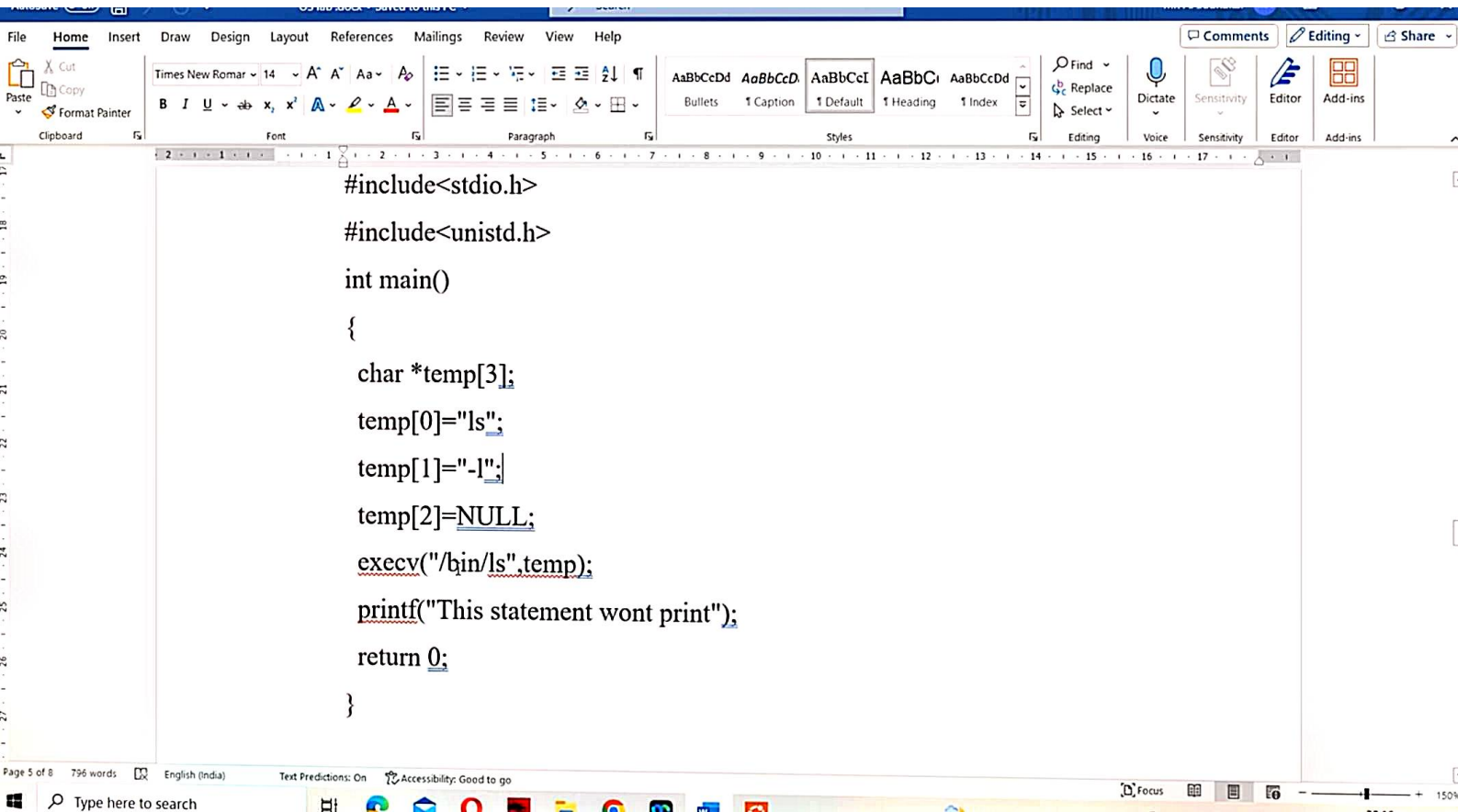
```
int main()
```

```
{
```

```
    char *temp[3];
```

```
    temp[0]="ls";
```

```
    temp[1]="-l";
```





Autosave... OS Word docx - Saved to this PC... Search... MPA - Student...

File Home Insert Draw Design Layout References Mailings Review View Help

Clipboard Font Paragraph Styles Editing Comments Share

Times New Roman 14 A<sup>+</sup> A<sup>-</sup> Aa Font Face, Size, Bold, Italic, Underline, Text Color, Background Color, Bullets, Paragraph, Styles, Find, Replace, Select, Dictate, Voice, Sensitivity, Editor, Add-ins

-rw-rw-r-- 1 ipc ipc 424 Feb 1 13:40 hi.c

---

**execlp() System Call:-**

```
#include<stdio.h>
#include<unistd.h>
int main()
{
    execlp("/bin/ls","ls",NULL);
    printf("This statement wont print");
    return 0;
}
```

Output:-|

Page 6 of 8 796 words English (India) Text Predictions: On Accessibility: Good to go

Type here to search

```
-rw-r--r-- 1 sudhakaratchala sudhakaratchala 176 Feb  2 20:05 ex3.c
-rw-r--r-- 1 sudhakaratchala sudhakaratchala 441 Feb  2 20:08 ex4.c
-rw-r--r-- 1 sudhakaratchala sudhakaratchala 477 Feb  2 20:14 ex5.c
-rw-r--r-- 1 sudhakaratchala sudhakaratchala 135 Feb  3 19:57 execlp.c
-rw-r--r-- 1 sudhakaratchala sudhakaratchala 194 Feb  3 19:56 execv.c
-rw-r--r-- 1 sudhakaratchala sudhakaratchala 301 Feb  2 11:43 exit.c
-rw-r--r-- 1 sudhakaratchala sudhakaratchala 125 Jan 28 15:31 fork.c
-rw-r--r-- 1 sudhakaratchala sudhakaratchala   4 Jan 28 10:35 hai
-rw-r--r-- 1 sudhakaratchala sudhakaratchala 173 Jan 28 15:41 hai.c
-rw-r--r-- 1 sudhakaratchala sudhakaratchala 137 Feb  1 12:08 p1.c
-rw-r--r-- 1 sudhakaratchala sudhakaratchala 423 Feb  2 11:36 sleep.c
```

sudhakaratchala@MSI:~\$ gcc execlp.c

sudhakaratchala@MSI:~\$ pico execlp.c

sudhakaratchala@MSI:~\$ gcc execlp.c

sudhakaratchala@MSI:~\$ ls

```
a.out  ex1.c  ex3.c  ex5.c      execv.c  fork.c  hai.c  sleep.c
```

```
ex.x   ex2.c  ex4.c  execlp.c  exit.c   hai     p1.c
```

sudhakaratchala@MSI:~\$ ./a.out

```
a.out  ex1.c  ex3.c  ex5.c      execv.c  fork.c  hai.c  sleep.c
```

```
ex.x   ex2.c  ex4.c  execlp.c  exit.c   hai     p1.c
```

sudhakaratchala@MSI:~\$

File Home Insert Draw Design Layout References Mailings Review View Help

Clipboard Font Paragraph Styles Editing Comments Share

Find Replace Select Dictate Voice Sensitivity Editor Add-ins

Page 7 of 8 796 words English (India) Text Predictions: On Accessibility: Good to go

Type here to search 26°C Clear ENG

#### 4. sleep() system call:-

It will keeps a running process in sleep state up to a specified time seconds. It allows a process to relinquish the CPU and enter a sleep state for a set period.

```
#include<stdio.h>
#include<unistd.h>
int main()
{
    int i,pid;
    pid=fork();
    if(pid==0)
    {
        printf("\"nchild process started\"");
        for(i=0;i<10;i++)
```

```
#include<stdio.h>
#include<unistd.h>
int main()
{
    int i,pid;
    pid=fork();
    if(pid==0)
    {
        printf("\nchild process started");
        for(i=0;i<10;i++)
        {
            sleep(1);
            printf("\ni=%d",i);
        }
        printf("\nchild process ends");
    }
    else
```

[ Read 23 lines ]

^G Help  
^X Exit

^O Write Out  
^R Read File


^W Where Is  
^\_ Replace

^K Cut  
^U Paste

^T Execute  
^J Justify

^C Location  
^/ Go To Line

Page 7 of 8 796 words English (India) Text Predictions: On Accessibility: Good to go



```
sudhakaratchala@MSI:~$ pico sleep.c
sudhakaratchala@MSI:~$ gcc sleep.c
sudhakaratchala@MSI:~$ ./a.out
```

```
I am parent
```

```
Parent process ends
sudhakaratchala@MSI:~$ child process started
i=0
```

### 5.exit() system call:-

- The exit system call is used in operating systems to terminate a process.
- It specifies that program has completed its execution and is ready to be terminated.
- The exit system call takes an exit status code, which is a small integer value that the process returns to the operating system.
- The status parameter is a value that is returned to the operating system. A zero status typically indicates successful execution, while non-zero values often represent error conditions.

`#include <stdio.h>`

`#include <stdlib.h>`



often represent error conditions.

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    printf("Before exit\n");
    exit(0);
    printf("After exit\n");
    return 0;
}
```

**Output:-**  
Before exit