OS Lab Exercise 2 -Implementation Of System Calls

Aim:

To write C Program to implement the following system calls

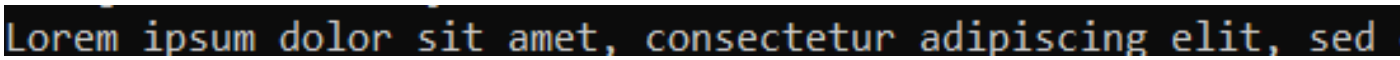
- Open ()
- Read ()
- Write ()
- Wait ()
- Exec ()
- Fork ()
- Sleep ()
- Getpid ()
- Lseek ()

Programs:

1 **Open:**

```
#include <sys/types.h>

#include <sys/stat.h>

#include <unistd.h>

#include <fcntl.h>

int main(){

int k=open("test.txt",O_RDONLY);

char buffer[100];

read(k,buffer,100);

write(1,buffer,100);

}
```

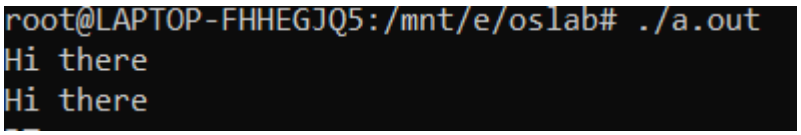
Output:



2. Read():

```
#include <stdio.h>

#include <unistd.h>

#include <sys/types.h>

int main(){

char buffer[15];

read(1,buffer,10);

printf("%s\n",buffer);

}
```

Output:



3.write():

```
#include <unistd.h>

int main(){

char bufferr[30]="hi there  hello world\n";

write(1,bufferr,30);

}
```

Output:

```
root@LAPTOP-FHHEGJQ5:/mnt/e/oslab# gcc write.c
root@LAPTOP-FHHEGJQ5:/mnt/e/oslab# ./a.out
hi there  hello world
```

4.wait():

```
#include <stdio.h>

#include <stdlib.h>

#include <sys/wait.h>

#include <unistd.h>

int main(){

        int status;

        if(fork()==0)

        {

        printf("Exiting..");

        exit(1);

        }else{

        wait(&status);

        }

        printf("Exit status %d",WEXITSTATUS(status));

}
```

Output:

```
root@LAPTOP-FHHEGJQ5:/mnt/e/oslab# gcc wait.c
root@LAPTOP-FHHEGJQ5:/mnt/e/oslab# ./a.out
Exiting..Exit status 1root@LAPTOP-FHHEGJQ5:/mnt/
```

5.Exec():

```
#include <unistd.h>

int main(){

char* path="/bin/ls";

char* arg[]={path,"-la",NULL};

execl(path,"-la",NULL);

execv(path,arg);

}
```

Output:

```
root@LAPTOP-FHHEGJQ5:/mnt/e/oslab# ./a.out
a.out    fork.c     fork2.c   oslabex1.docx   oslabex2.docx   read.c   sleep.c   wait.c   '~$labex2.docx'
exec.c   fork.out   open.c    oslabex1.pdf    pid.c           seek.c   test.txt  write.c
```

6. Fork():

```
#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

void test(){

fork();
```

```
//fork();

//fork();

printf("test\n");

}

int main(){

test();

//fork()&&fork()||fork();

//fork();

//fork();

//fork();

printf("hello\n");

}
```

Output:

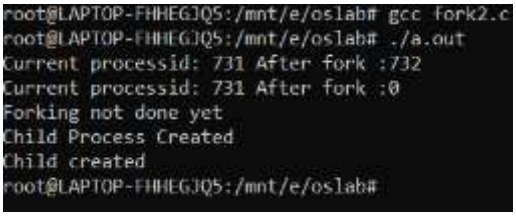
```
root@LAPTOP-FHHEGJQ5:/mnt/e/oslab# gcc fork.c
root@LAPTOP-FHHEGJQ5:/mnt/e/oslab# ./a.out
test
test
hello
hello
```

Fork():

```
#include <sys/types.h>

#include <stdio.h>

#include <unistd.h>
```

```
#include <sys/wait.h>

int main(){

printf("Current processid: %d",(int)getpid());

pid_t a=fork();

printf(" After fork :%d\n",(int)a);

if(a<0){

fprintf(stderr,"Error \n");



}

else if (a==0){

printf("Child Process Created\n");

}

else{

printf("Forking not done yet\n");

wait(NULL);

printf("Child created\n");

}

}
```

Output:



7.sleep():

```
#include <stdio.h>

#include <unistd.h>

int main(){

printf("Hi there:\n");

sleep(3);

printf("hello\n");

}
```

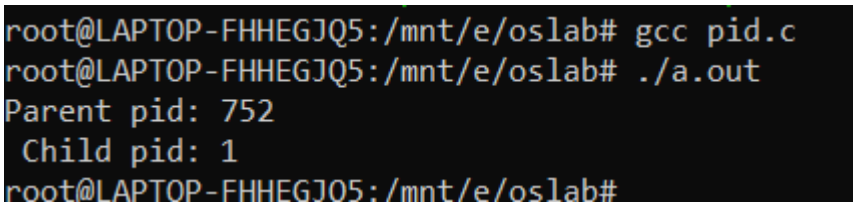
Output:

```
root@LAPTOP-FHHEGJQ5:/mnt/e/oslab# gcc sleep.c
root@LAPTOP-FHHEGJQ5:/mnt/e/oslab# ./a.out
Hi there:
hello
```

8.getpid():

```
#include <unistd.h>

#include <stdio.h>

int main(){

if(fork()==0){

printf("Parent pid: %d\n ",getpid());

printf("Child pid: %d\n",getppid());

}

}
```

Output:



9.lseek():

```
#include <sys/stat.h>

#include <sys/types.h>

#include <fcntl.h>

#include <unistd.h>

int main(){

int buf[40];

int fd=open("test.txt",O_RDWR);

read(fd,buf,40);

write(1,buf,40);

lseek(fd,15,SEEK_SET);

write(1,"\n",1);

read(fd,buf,40);

write(1,buf,40);

}
```

Output:

```
root@LAPTOP-FHHEGJQ5:/mnt/e/oslab# gcc seek.c
root@LAPTOP-FHHEGJQ5:/mnt/e/oslab# ./a.out
Lorem ipsum dolor sit amet, consectetur
or sit amet, consectetur adipiscing elitroot@L
```

Test.txt:

```
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna ali
ua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis au
e irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat
upidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
```

Result:

Thus the above linux system calls were implemented in c