

BCSE303P- Operating Systems Lab

ASSESSMENT-2

File Operations and fork()
system call

Name: Santhosh Kumar S

Regno: 21BCE1829

Slot: L15+L16

Faculty: DR. Anandan P

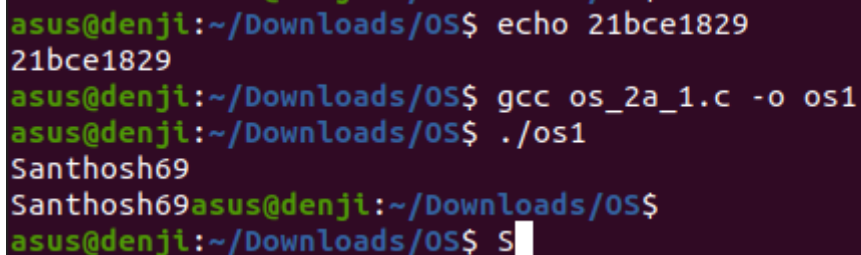
2 a (File operation)

1. Develop a C program to get the input from standard input device and print the same on the standard output device.

Code:

```
#include <stdio.h>
#include <unistd.h>
int main(){
    int count,count1;
    char buff[20];
    count = read(0,buff,10);
    count=write(1,buff,10);
}
```

Output:

A terminal window with a dark purple background. The prompt is 'asus@denji:~/Downloads/OS\$'. The user enters 'echo 21bce1829' and the output is '21bce1829'. Then the user enters 'gcc os_2a_1.c -o os1' and the prompt returns. Next, the user enters './os1' and the output is 'Santhosh69'. Finally, the user enters 'S' and the prompt returns.

```
asus@denji:~/Downloads/OS$ echo 21bce1829
21bce1829
asus@denji:~/Downloads/OS$ gcc os_2a_1.c -o os1
asus@denji:~/Downloads/OS$ ./os1
Santhosh69
Santhosh69asus@denji:~/Downloads/OS$
asus@denji:~/Downloads/OS$ S
```

2. Develop a C program to perform the following file operations – Create , Write, Read, Copy.

Code:

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

int main(){
    char buff[11];
    int fd= open("foo.txt",O_WRONLY|O_CREAT);
    int c= write(fd,"Hello-World",11);
    fd=open("foo.txt",O_RDONLY);
    read(fd,buff,c);
    printf("Message in the file foo.txt is :%s\n",buff);
    close(fd);
    fd=open("foo.txt",O_RDONLY);
    read(fd,buff,c);
    close(fd);
    int fd1=open("foo1.txt",O_WRONLY|O_CREAT);
    write(fd1,buff,c);
    close(fd1);
    fd=open("foo.txt",O_RDONLY);
    read(fd,buff,c);
    printf("Message written in new file: %s",buff);
}
```

Output:

```
asus@denji:~/Downloads/OS$ gcc 21bce1829_ass3.c -o os3
asus@denji:~/Downloads/OS$ ./os3
Message in the file foo.txt is :Hello-WorldVU
Message written in new file: Hello-WorldVU
asus@denji:~/Downloads/OS$ gcc 21bce1829_ass3.c -o os3
asus@denji:~/Downloads/OS$ ./os3
Message in the file foo.txt is :Hello-World
Message written in new file: Hello-World
asus@denji:~/Downloads/OS$ echo 21bce1829
21bce1829
asus@denji:~/Downloads/OS$ echo 21bce1829
21bce1829
asus@denji:~/Downloads/OS$
```

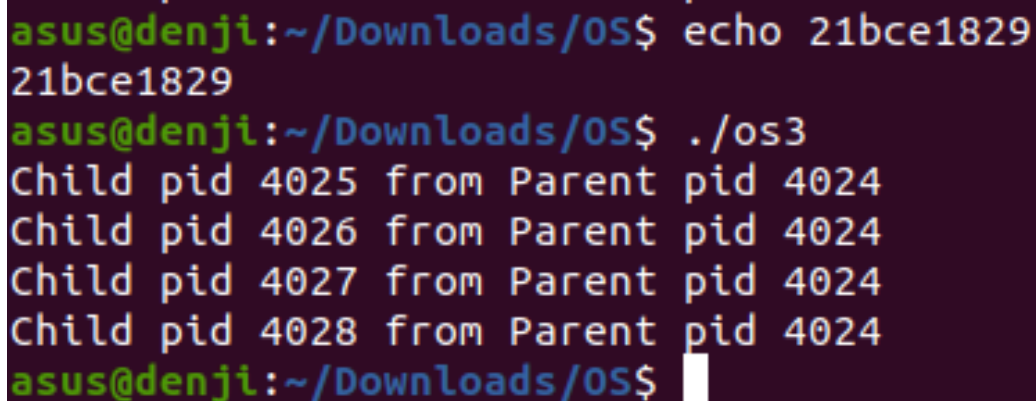
2 b) (Fork() system call)

3. Develop a C program to create four child processes. Print the parent ID in all child processes along with its own ID. Print all Child IDs in parent process along with its own ID.

Code:

```
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>
#include<stdio.h>
int main()
{
    for(int i=0;i<4;i++)
    {
        if(fork() == 0)
        {
            printf("Child pid %d from Parent pid %d\n",getpid(),getppid());
        }
    }
}
```

Output:



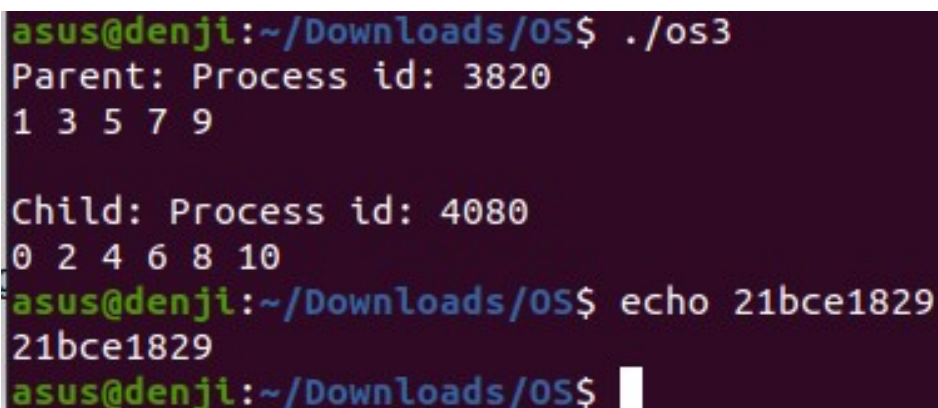
```
asus@denji:~/Downloads/OS$ echo 21bce1829
21bce1829
asus@denji:~/Downloads/OS$ ./os3
Child pid 4025 from Parent pid 4024
Child pid 4026 from Parent pid 4024
Child pid 4027 from Parent pid 4024
Child pid 4028 from Parent pid 4024
asus@denji:~/Downloads/OS$
```

4. Develop a C program to print even numbers between 0 and 10 within Child process & print odd numbers between 0 and 10 within Parent process.

Code:

```
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>
#include<stdio.h>
int main()
{
    int pid;
    pid=fork();
    if(pid==0)
    {
        printf("Child: Process id: %d\n",getpid());
        for(int i=0;i<=10;i++)
        {
            (i%2==0)?printf("%d ",i):printf("");
        }
        printf("\n");
    }
    else if(pid>0)
    {
        printf("Parent: Process id: %d\n",getppid());
        for(int i=0;i<=10;i++)
        {
            (i%2!=0)?printf("%d ",i):printf("");
        }
        printf("\n\n");
    }
    return 0;
}
```

Output:



```
asus@denji:~/Downloads/OS$ ./os3
Parent: Process id: 3820
1 3 5 7 9

Child: Process id: 4080
0 2 4 6 8 10
asus@denji:~/Downloads/OS$ echo 21bce1829
21bce1829
asus@denji:~/Downloads/OS$
```

5. Develop a C program to perform 2x2 matrix addition in child process and determine whether the given number is prime or not in parent process.

Code:

```
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>
#include<stdio.h>
void parent()
{
    int flag=1,n;
    scanf("%d",&n);
    for(int i=1;i<n/2;i++)
    {
        if(n%i==0)
        {
            flag=0;
            break;
        }
    }
    (flag==1)?printf("The number given is a prime number\n"):printf("The number
given is not a prime number\n");
}
void child()
{
    int a[2][2],b[2][2],sum[2][2];
    for(int i=0;i<2;i++)
    {
        for(int j=0;j<2;j++)
        {
            scanf("%d",&a[i][j]);
        }
    }
    for(int i=0;i<2;i++)
    {
        for(int j=0;j<2;j++)
        {
            scanf("%d",&b[i][j]);
        }
    }
    for(int i=0;i<2;i++)
    {
        for(int j=0;j<2;j++)
        {
            sum[i][j]=a[i][j]+b[i][j];
        }
    }
    for(int i=0;i<2;i++)
    {
        for(int j=0;j<2;j++)
        {
```

```

        printf("%d\t",sum[i][j]);
    }
    printf("\n");
}
}
int main()
{
    int pid;
    pid=fork();
    if(pid==0)
    {
        child();
    }

    if(pid>0)
    {
        parent();
        printf("Parent: Process id: %d\n\n",getppid());
    }else{
        printf("Child: Process id: %d\n",getpid());
    }
    if(pid>0)
    {
        parent();
        printf("Parent: Process id: %d\n\n",getppid());
    }
    return 0;}

```

Output:

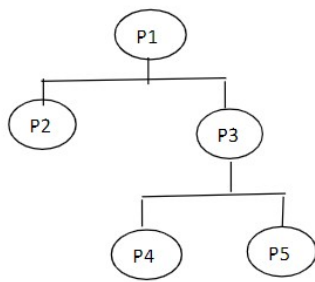
```

asus@denji:~/Downloads/OS$ ./os3
234
The number given is not a prime number
Parent: Process id: 3820

1 2 3 4 5 6 7 8 9
6      8
10     12
Child: Process id: 4389
^C
asus@denji:~/Downloads/OS$ echo 21bce1829
21bce1829
asus@denji:~/Downloads/OS$ █

```


6. Develop a C program to create a process hierarchy as shown below.



Code:

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main() {
    int P1,P2,P3,P4,P5;
    int pid1=fork();
    if(pid1==0)
    {
        printf("P2 of child process id %d under %d\n",getpid(),getppid());
    }
    else
    {
        int pid2=fork();
        if(pid2==0)
        {
            int pid3=fork();
            if(pid3==0)
            {
                printf("P4 of child process id %d under %d\n",getpid(),getppid());
            }
            else
            {
                int pid4=fork();
                if(pid4==0)
                {
                    printf("P5 of child process id %d under %d\n",getpid(),getppid());
                }
                else
                {
                    printf("P3 of child process id %d under %d\n",getpid(),getppid());
                }
            }
        }
    }
    else
    {
        printf("P1 parent process id :%d\n",getpid());
    }
}
return 0;}
```

Output:

```
asus@denji:~/Downloads/OS$ ./os3
P1 parent process id :4490
P2 of child process id 4491 under 4490
P4 of child process id 4493 under 4492
P3 of child process id 4492 under 4490
asus@denji:~/Downloads/OS$ P5 of child process id 4494 under 4492
echo 21bce1829
21bce1829
asus@denji:~/Downloads/OS$
```