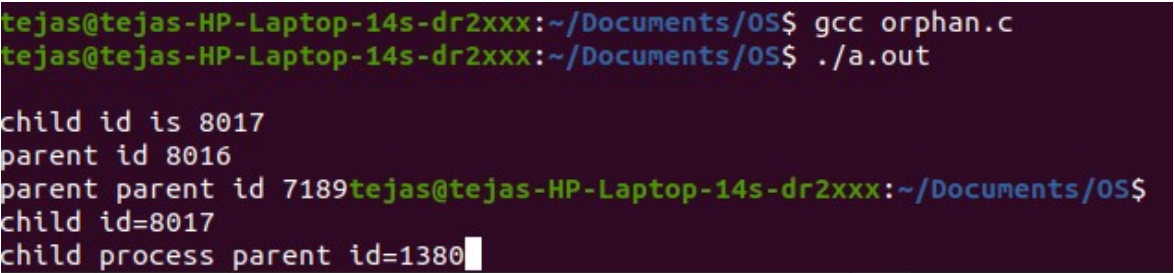


Orphan State:

```
# include<sys/types.h>
# include<sys/wait.h>
# include<unistd.h>
# include<stdio.h>
#include<stdlib.h>
void main()
{
int status;
pid_t pid;
pid=fork();
if(pid==0)
{
sleep(2);
pid=getpid();
printf("\nchild id=%d",pid);
pid=getppid();
printf("\nchild process parent id=%d",pid);
}
else
{
printf("\nchild id is %d",pid);
pid=getpid();
printf("\nparent id %d",pid);
pid=getppid();
printf("\nparent parent id %d",pid);
}
}
```

Output:



```
tejas@tejas-HP-Laptop-14s-dr2xxx:~/Documents/OS$ gcc orphan.c
tejas@tejas-HP-Laptop-14s-dr2xxx:~/Documents/OS$ ./a.out

child id is 8017
parent id 8016
parent parent id 7189tejas@tejas-HP-Laptop-14s-dr2xxx:~/Documents/OS$
child id=8017
child process parent id=1380
```

Zombie State:

```
# include<sys/types.h>
# include<sys/wait.h>
# include<unistd.h>
# include<stdio.h>
#include<stdlib.h>
void main()
{
int status;
pid_t pid;
pid=fork();
if(pid==0)
{
pid=getpid();
printf("\nchild id=%d",pid);
pid=getppid();
printf("\nchild process parent id=%d",pid);
}
else
{
sleep(2);
system("ps");
wait(&status);
system("ps");
printf("\nchild id is %d",pid);
pid=getpid();
printf("\nparent id %d",pid);
pid=getppid();
printf("\nparent parent id %d",pid);
}
}
```

Output:

```
tejas@tejas-HP-Laptop-14s-dr2xxx:~/Documents/OS$ gcc zombie.c
tejas@tejas-HP-Laptop-14s-dr2xxx:~/Documents/OS$ ./a.out
```

```
child id=8161
child process parent id=8160      PID TTY          TIME CMD
 7189 pts/1        00:00:00 bash
 8160 pts/1        00:00:00 a.out
 8161 pts/1        00:00:00 a.out <defunct>
 8162 pts/1        00:00:00 sh
 8163 pts/1        00:00:00 ps
  PID TTY          TIME CMD
 7189 pts/1        00:00:00 bash
 8160 pts/1        00:00:00 a.out
 8164 pts/1        00:00:00 sh
 8165 pts/1        00:00:00 ps
```

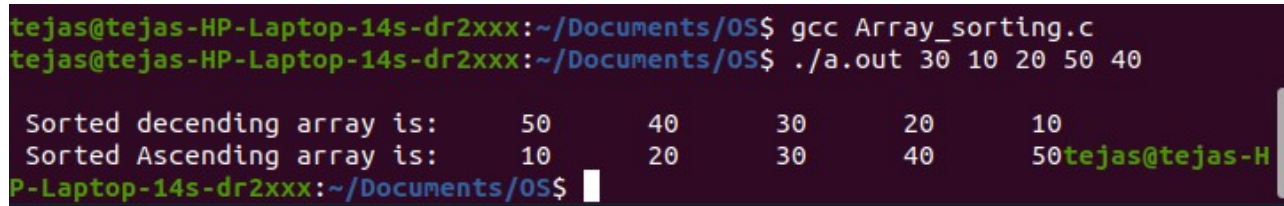
```
child id is 8161
parent id 8160
parent parent id 7189tejas@tejas-HP-Laptop-14s-dr2xxx:~/Documents/OS$
```

Array Sorting

```
# include<sys/types.h>
# include<unistd.h>
# include<stdio.h>
# include<stdlib.h>
void main(int argc, char *argv[])
{
    pid_t pid;
    int n;
    int temp;
    int arr[10];
    /*printf("\nEnter the size of array:");
    scanf("%d",&n);*/
    for(int i=1;i<argc;i++)
    {
        arr[i-1]=atoi(argv[i]);
    }
    n=argc-1;
    pid=fork();
    if(pid==0)
    {
        for(int i=0;i<n;i++)
        {
            for(int j=0;j<n-1;j++)
            {
                if(arr[j]>arr[j+1])
                {
                    temp=arr[j];
                    arr[j]=arr[j+1];
                    arr[j+1]=temp;
                }
            }
        }
        printf("\n Sorted Ascending array is:");
        for(int i=0;i<n;i++)
        {
            printf("\t%d",arr[i]);
        }
    }
    else
    {
        for(int i=0;i<n;i++)
        {
            for(int j=0;j<n-1;j++)
            {
                if(arr[j]<arr[j+1])
                {
                    temp=arr[j];
                    arr[j]=arr[j+1];
                    arr[j+1]=temp;
                }
            }
        }
    }
}
```

```
}  
}  
printf("\n Sorted decending array is:");  
for(int i=0;i<n;i++)  
{  
    printf("\t%d",arr[i]);  
}  
}  
}
```

Output:



A terminal window screenshot showing the compilation and execution of a C program. The user is at a prompt in the directory ~/Documents/OS. They compile the file Array_sorting.c using gcc, then run the resulting executable ./a.out with arguments 30 10 20 50 40. The program outputs two lines: 'Sorted decending array is:' followed by the numbers 50, 40, 30, 20, 10, and 'Sorted Ascending array is:' followed by 10, 20, 30, 40, 50. The terminal text is as follows:

```
tejas@tejas-HP-Laptop-14s-dr2xxx:~/Documents/OS$ gcc Array_sorting.c  
tejas@tejas-HP-Laptop-14s-dr2xxx:~/Documents/OS$ ./a.out 30 10 20 50 40  
  
Sorted decending array is:      50      40      30      20      10  
Sorted Ascending array is:     10      20      30      40      50tejas@tejas-H  
P-Laptop-14s-dr2xxx:~/Documents/OS$
```