

✓ Program Name

Sorting using Parent and Child Process (fork system call)

The parent process sorts the array in **descending order**, while the child process sorts it in **ascending order**.

✓ Concept / Theory (For Practical File)

What is fork()?

- fork() is a system call used in UNIX/Linux to create a **new process**.
- The new process is called the **child process**.
- The process that calls fork() is the **parent process**.
- After fork(), **both processes run the same program**, but have different **Process IDs (PIDs)**.

Return values of fork()

Return Value Process

0	Child Process
Positive PID	Parent Process
-1	Error (fork failed)

Why use fork in this program?

- To demonstrate **process creation** and **parallel execution**.
 - The **child process** sorts the same array **ascending**, and the **parent process** sorts it **descending**.
 - Both processes have **separate copies** of the array in memory (because child gets a copy of parent's memory).
-

✓ Algorithm

1. Start.
 2. Accept size n and array elements from user.
 3. Call fork() to create a child process.
 4. If fork() returns 0, it is **child** → perform **ascending sort** and print PID.
 5. If fork() returns positive value, it is **parent** → perform **descending sort** and print PID.
 6. End the program.
-

✓ Line-by-Line Explanation (Short Viva Notes)

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

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

- stdio.h for input/output functions like printf, scanf.
 - unistd.h contains the fork() and getpid() functions.
-

```
void sortAsc(int a[], int n) {
```

- Function to sort array in **ascending order** (child process).
-

```
for(int i=0;i<n;i++)
```

```
for(int j=i+1;j<n;j++)
```

```
if(a[i] > a[j]) { int t=a[i]; a[i]=a[j]; a[j]=t; }
```

- Simple **bubble-style sorting logic** (swap if bigger).
-

```
printf("\n[Child | PID %d] Sorted in Ascending Order: ", getpid());
```

- getpid() prints the **process ID of the child**.
-

```
void sortDesc(int a[], int n) {
```

- Function to sort array in **descending order** (parent process).
-

```
if(a[i] < a[j]) { int t=a[i]; a[i]=a[j]; a[j]=t; }
```

- Swap if smaller → descending sort.
-

```
pid_t pid = fork();
```

- Creates child process.
-

```
if(pid == 0) {
```

```
    sortAsc(a, n); // Child process
```

```
} else {
```

```
    sortDesc(a, n); // Parent process
```

```
}
```

- pid == 0 → Child executes ascending sort
- pid > 0 → Parent executes descending sort

 **Sample Output (Write in Journal)**

Enter size: 5

Enter elements: 10 5 3 8 2

[Child | PID 3456] Sorted in Ascending Order: 2 3 5 8 10

[Parent | PID 3455] Sorted in Descending Order: 10 8 5 3 2