```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>

// Function for Bubble Sort
void bubbleSort(int arr[], int n) {
    int i, j, temp;
    for(i = 0; i < n-1; i++) {
        for(j = 0; j < n-i-1; j++) {
            if(arr[j] > arr[j+1]) {
                temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }
}

// Function for Selection Sort
void selectionSort(int arr[], int n) {
    int i, j, min, temp;
    for(i = 0; i < n-1; i++) {
        min = i;
        for(j = i+1; j < n; j++) {
            if(arr[j] < arr[min])
                min = j;
        }
        temp = arr[min];
        arr[min] = arr[i];
        arr[i] = temp;
    }
}

int main() {
    int n, i;
    printf("Enter number of integers: ");
    scanf("%d", &n);

    int arr[n];
    printf("Enter %d integers: ", n);
    for(i = 0; i < n; i++)
        scanf("%d", &arr[i]);

    pid_t pid = fork();

    if(pid < 0) {
        printf("Fork failed!\n");
        exit(1);
    }

    else if(pid == 0) { // Child process
        printf("\nChild Process started (PID: %d, PPID: %d)\n", getpid(), getppid());
        printf("Sorting using Selection Sort...\n");
        selectionSort(arr, n);

        printf("Child Process Sorted Array: ");
        for(i = 0; i < n; i++)
            printf("%d ", arr[i]);
        printf("\n");

        printf("\nChild sleeping for 10 seconds...\n");
        sleep(10); // Parent exits before child finishes
        printf("\nAfter 10 seconds, Child’s new Parent PID: %d\n", getppid());
        printf("Child process continuing (Orphan demonstrated).\n");
    }

    else { // Parent process
        printf("\nParent Process (PID: %d), Child PID: %d\n", getpid(), pid);
        printf("Sorting using Bubble Sort...\n");
        bubbleSort(arr, n);

        printf("Parent Process Sorted Array: ");
        for(i = 0; i < n; i++)
            printf("%d ", arr[i]);
        printf("\n");

        printf("\nParent exiting before child completes...\n");
        exit(0); // Parent terminates early — child becomes orphan
    }

    return 0;
}
```

```
nano orphan_process.c
gcc orphan_process.c -o orphan_process
./orphan_process
```

input

```
Enter number of integers: 5
Enter 5 integers: 9 4 2 8 6
```

output

```
Parent Process (PID: 5212), Child PID: 5213
Sorting using Bubble Sort...
Parent Process Sorted Array: 2 4 6 8 9

Parent exiting before child completes...

Child Process started (PID: 5213, PPID: 5212)
Sorting using Selection Sort...
Child Process Sorted Array: 2 4 6 8 9

Child sleeping for 10 seconds...

After 10 seconds, Child’s new Parent PID: 1
Child process continuing (Orphan demonstrated).
```