FORKFINAL.C

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

// Function to perform bubble sort
void bubble_sort(int arr[], int n) {
    int temp;
    for (int i = 0; i < n - 1; i++) {
        for (int 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 to display the array
void display_array(int arr[], int n) {
    printf("Array: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

int main() {
    int n;

    // Ask the user for the size of the array
    printf("Enter the number of elements in the array: ");
    scanf("%d", &n);

    int arr[n];  // Declare array of size n

    // Get array elements from the user
    printf("Enter the elements of the array: ");
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    printf("Original Array:\n");
    display_array(arr, n);

    pid_t pid = fork(); // Create a child process
    if (pid < 0) {
        // Error in fork
        perror("Fork failed");
        return 1;
    }
    else if (pid == 0) {
        // Child process
        // The child will sort the array again, and show how the state becomes orphaned
        printf("Child Process (PID: %d): I will sort the array again.\n", getpid());

        sleep(5);  // Simulate a delay to let the parent process exit first

        bubble_sort(arr, n);  // Child sorts the array
        printf("Child Process (PID: %d): Sorted Array (Again) by child: ", getpid());
        display_array(arr, n);

        printf("Child Process: I am exiting now.\n");
        exit(0);  // Child exits, and will become a zombie state after the parent exits
    }
    else {
        // Parent process
        // The parent sorts the array first
        printf("Parent Process (PID: %d): Sorting the array first.\n", getpid());
        bubble_sort(arr, n);  // Parent sorts the array
        display_array(arr, n);  // Display sorted array

        printf("Parent Process: I am about to exit and will leave the child process as a Zombie.\n");

        // Parent calls wait() to wait for the child to finish
        wait(NULL);  // Wait for child to finish execution

        printf("Parent Process: The child process has finished.\n");

        // Parent exits, leaving the child process in the orphan state
        sleep(2);  // Sleep to allow the child process to run after parent exits
        printf("Parent Process (PID: %d): I have exited, leaving my child as an orphan process.\n",
getpid());
        exit(0);  // Parent exits
    }

    return 0;
}
```