Parallel Array Sum with Fork

Objective

Write a C program that calculates the sum of an array of integers by dividing the workload between multiple processes created using the `fork()` function.

Instructions

1. **Initialize an array:**
   - Create an array of 20 integers. Initialize it with random values between 1 and 100.
2. **Determine process count:**
   - Divide the work between 4 child processes. Each process should calculate the sum of a specific portion of the array.
3. **Use `fork()`:**
   - Use `fork()` to create 4 child processes.
   - Each child process should handle a unique portion of the array (for example, the first 5 elements for the first process, the next 5 for the second, etc.).
4. **Shared memory segment:**
   - Create a shared memory segment where each child process can store its calculated sum. Use `shmget()` and `shmat()` for shared memory management.
   - Ensure each process writes its partial sum to a unique position in the shared memory segment.
5. **Parent process:**
   - After creating the child processes, the parent process should wait for all child processes to complete using `wait()`.
   - Once all child processes have completed, the parent should read the partial sums from the shared memory and compute the total sum of the array.
6. **Output:**
   - Display the partial sum computed by each child process.
   - Display the total sum computed by the parent process.

Bonus Challenge

To make the exercise harder, you can:

- Implement error handling for system calls.
- Use `pipe()` to send completion status from each child process back to the parent.
- Measure and output the time taken to compute the sum using multiple processes versus a single process.

Example Output

```
Child 1 calculated partial sum: 100
Child 2 calculated partial sum: 150
Child 3 calculated partial sum: 120
Child 4 calculated partial sum: 130
Total sum calculated by parent: 500
```