Name : Sarthak Pagar

Roll No. : 34

Classs : TE (IT)

Practical : 2B

Statement : [Implement the C program in which main program accepts an array. Main program uses the FORK system call to create a new process called a child process. Parent process sorts an array and passes the sorted array to child process through the command line arguments of EXECVE system call. The child process uses EXECVE system call to load new program which display array in reverse order.](https://drive.google.com/drive/folders/1oX-lhryOFVy3sKXIBrwF4Te0cf1YAUK2?usp=sharing)

Parent.c :-

#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

#include <sys/wait.h>

#include <stdlib.h>

void bubble\_asc(int arr[], int n) {

int i, j, temp;

for (i = 0; i < n; 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;

}

}

}

printf("\nArray in ascending order: ");

for (i = 0; i < n; i++) {

printf("\t%d", arr[i]);

}

printf("\n");

}

int main(int argc, char \*argv[]) {

if (argc < 2) {

printf("Please provide numbers to sort.\n");

return 1;

}

int n = argc - 1; // Number of integers provided

int a[n];

pid\_t pid;

// Convert arguments to integers

for (int i = 1; i <= n; i++) {

a[i - 1] = atoi(argv[i]); // Store the integers in array

}

bubble\_asc(a, n); // Sort in ascending order

// Prepare arguments for child process

char \*args[n + 2]; // +2 for program name and NULL termination

args[0] = "./child"; // Assume the child executable is named "child"

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

args[i + 1] = argv[i + 1]; // Pass sorted values

}

args[n + 1] = NULL; // NULL terminate the argument list

pid = fork();

if (pid == 0) { // Child process

printf("\nI am Child Process, my pid is %d \n", getpid());

execve("./child", args, NULL); // Execute child program

perror("execve failed"); // Only reached if execve fails

exit(1);

} else if (pid > 0) { // Parent process

wait(NULL); // Wait for child process to finish

printf("\nI am Parent Process, my pid is %d \n", getpid());

} else {

perror("Fork failed");

return 1;

}

return 0;

}

Child.c :-

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <unistd.h>

void bubble\_dsc(int arr[], int n) {

int i, j, temp;

for (i = 0; i < n; 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;

}

}

}

printf("\nArray in descending order: ");

for (i = 0; i < n; i++) {

printf("\t%d", arr[i]);

}

printf("\n");

}

int main(int argc, char \*argv[]) {

if (argc < 2) {

printf("No numbers provided.\n");

return 1;

}

int n = argc - 1; // Number of integers

int a[n];

// Convert arguments to integers

for (int i = 1; i <= n; i++) {

a[i - 1] = atoi(argv[i]); // Store the integers in array

}

bubble\_dsc(a, n); // Sort in descending order

return 0;

}

Output :-

[Saru1594@localhost 2B]$ gcc -o parent parent.c

[Saru1594@localhost 2B]$ gcc -o child child.c

[Saru1594@localhost 2B]$ ./parent 6 2 4 9 3 1 0

Array in ascending order: 0 1 2 3 4 6 9

I am Child Process, my pid is 12476

Array in descending order: 9 6 4 3 2 1 0

I am Parent Process, my pid is 12475