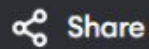


main.c



Share

Run

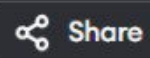
Output

```
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <sys/types.h>
5
6 int main() {
7     pid_t pid, ppid;
8
9     pid = fork();
10
11     if (pid < 0) {
12         perror("Fork failed");
13         exit(EXIT_FAILURE);
14     } else if (pid == 0) {
15
16         ppid = getppid();
17         printf("Child Process ID: %d\n", getpid());
18         printf("Parent Process ID: %d\n", ppid);
19     } else {
20
21         printf("Parent Process ID: %d\n", getpid());
22         printf("Created Child Process ID: %d\n", pid);
23     }
24
25     return 0;
26 }
27
```

Parent Process ID: 9210
Created Child Process ID: 9211
Child Process ID: 9211
Parent Process ID: 9210

=== Code Execution Successful ===

main.c



Run

Output

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <fcntl.h>
4 #include <unistd.h>
5 #define BUFFER_SIZE 1024
6 int main(int argc, char *argv[]) {
7     int source, destination, bytesRead;
8     char buffer[BUFFER_SIZE];
9     if (argc != 3) {
10         write(2, "Usage: ./filecopy <source> <destination>\n", 42);
11         exit(1);
12     }
13     source = open(argv[1], O_RDONLY);
14     if (source < 0) {
15         perror("Error opening source file");
16         exit(1);
17     }
18     destination = open(argv[2], O_WRONLY | O_CREAT | O_TRUNC, 0644);
19     if (destination < 0) {
20         perror("Error opening destination file");
21         close(source);
22         exit(1);
23     }
24     while ((bytesRead = read(source, buffer, BUFFER_SIZE)) > 0) {
25         write(destination, buffer, bytesRead);
26     }
27     close(source);
```

Usage: ./filecopy <source> <destination>

.

=== Code Exited With Errors ===