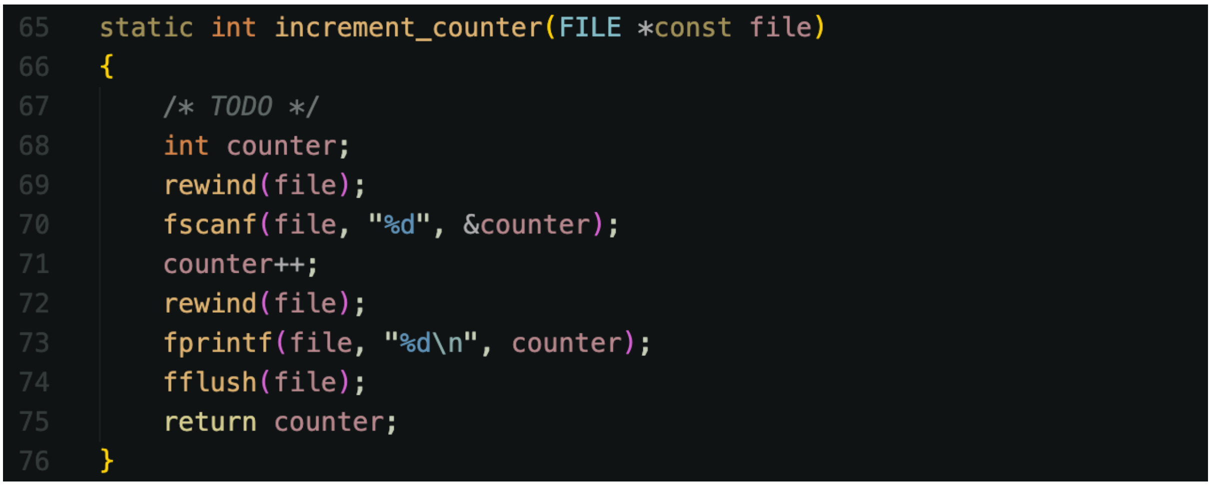
Unix Assignment8

Team19 108072244 邱煒甯 108032053 陳凱揚 112065525 簡佩如

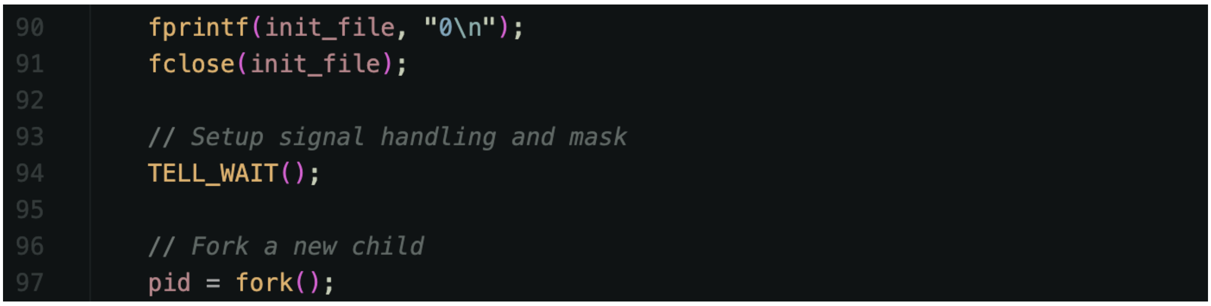
Implementation

In the `increment_counter` function, we set the file position to the beginning of the file using `rewind()`, then we read the current counter value, increase it, and overwrite the updated value back to the beginning of the file.



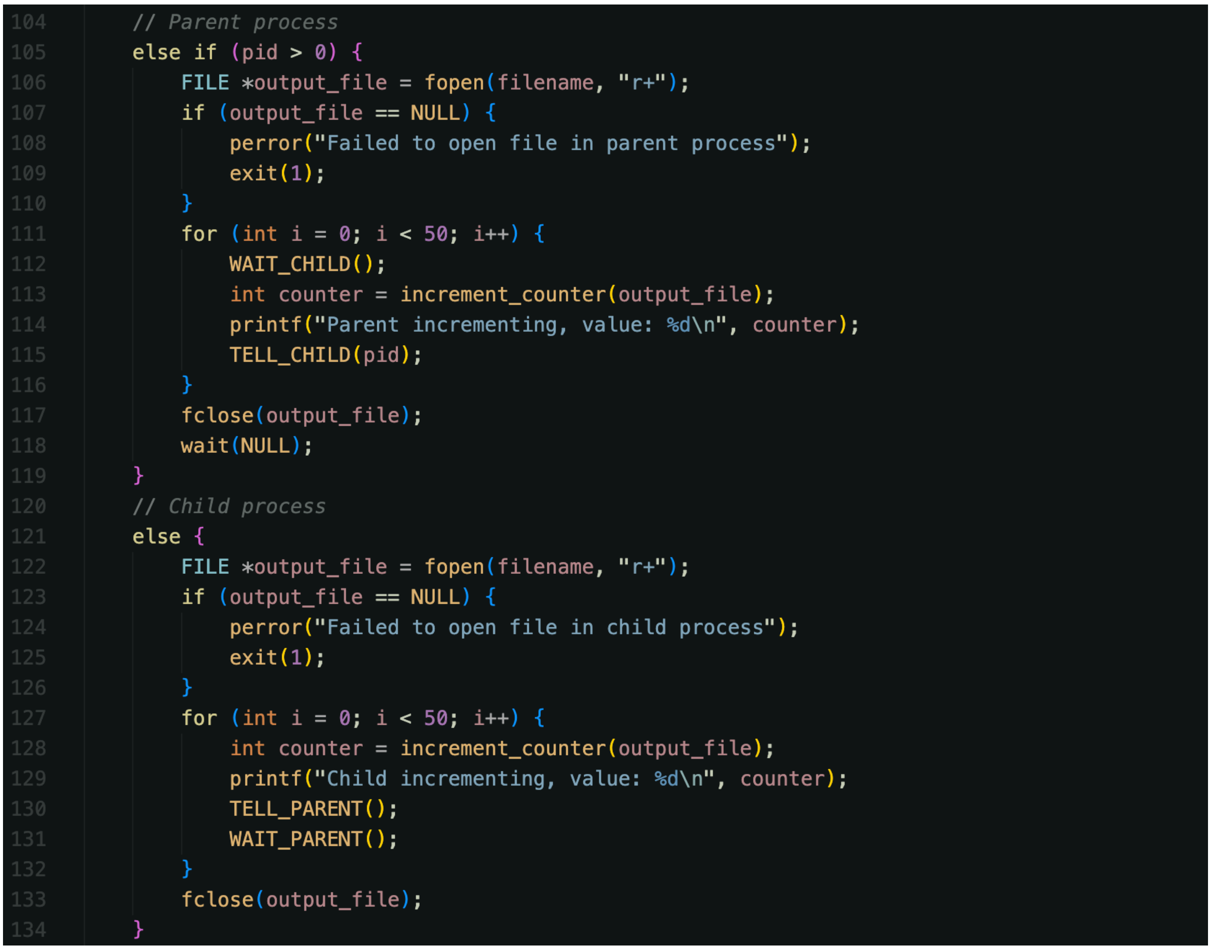
```c
static int increment_counter(FILE *const file)
{
    /* TODO */
    int counter;
    rewind(file);
    fscanf(file, "%d", &counter);
    counter++;
    rewind(file);
    fprintf(file, "%d\n", counter);
    fflush(file);
    return counter;
}
```

In the `main` function, we first create an output file and write the value "0" to it. Subsequently, we call `TELL_WAIT()` function to setup signal handling and mask, and then proceed to create a child process using `fork()`.



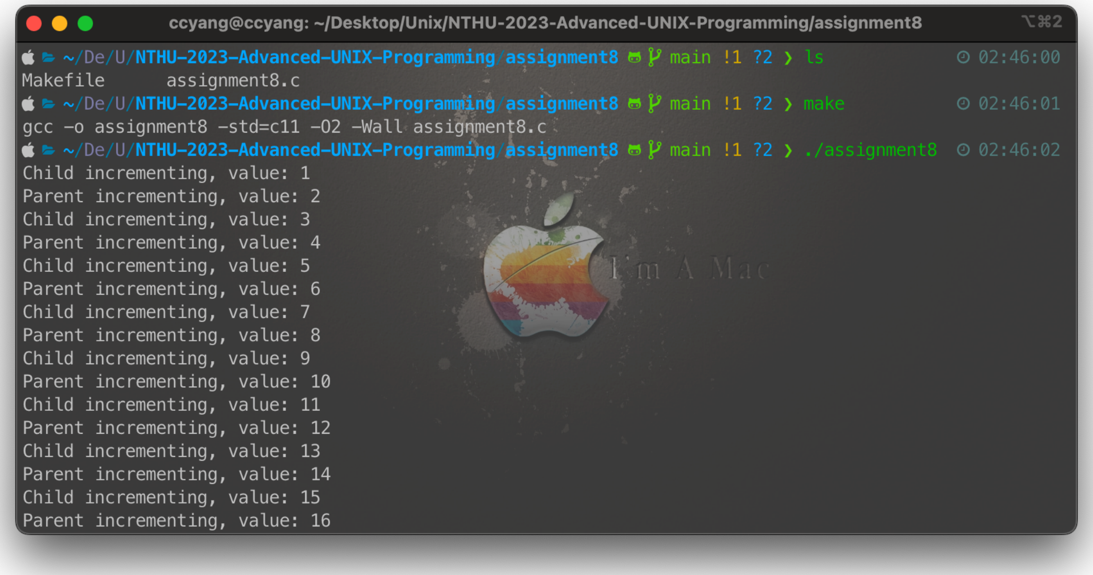
```c
    fprintf(init_file, "0\n");
    fclose(init_file);

    // Setup signal handling and mask
    TELL_WAIT();

    // Fork a new child
    pid = fork();
```

In both the parent and child processes, we open the output file and call `increment_counter()` function to increase the counter 50 times. The difference is the placement of the `WAIT()` function. In the parent process, `WAIT_CHILD()` is called at the start of each iteration. Conversely, in the child process, `WAIT_PARENT()` is called at the end of each iteration. This arrangement ensures parent and child process take turn incrementing the counter, starting from the child process.



```c
    // Parent process
    else if (pid > 0) {
        FILE *output_file = fopen(filename, "r+");
        if (output_file == NULL) {
            perror("Failed to open file in parent process");
            exit(1);
        }
        for (int i = 0; i < 50; i++) {
            WAIT_CHILD();
            int counter = increment_counter(output_file);
            printf("Parent incrementing, value: %d\n", counter);
            TELL_CHILD(pid);
        }
        fclose(output_file);
        wait(NULL);
    }
    // Child process
    else {
        FILE *output_file = fopen(filename, "r+");
        if (output_file == NULL) {
            perror("Failed to open file in child process");
            exit(1);
        }
        for (int i = 0; i < 50; i++) {
            int counter = increment_counter(output_file);
            printf("Child incrementing, value: %d\n", counter);
            TELL_PARENT();
            WAIT_PARENT();
        }
        fclose(output_file);
    }
```

Result



ccyang@ccyang: ~/Desktop/Unix/NTHU-2023-Advanced-UNIX-Programming/assignment8
⌥⌘2
I'm A Mac

```
~/De/U/NTHU-2023-Advanced-UNIX-Programming/assignment8  main !1 ?2 ❯ ls              02:46:00
Makefile      assignment8.c
~/De/U/NTHU-2023-Advanced-UNIX-Programming/assignment8  main !1 ?2 ❯ make            02:46:01
gcc -o assignment8 -std=c11 -O2 -Wall assignment8.c
~/De/U/NTHU-2023-Advanced-UNIX-Programming/assignment8  main !1 ?2 ❯ ./assignment8   02:46:02
Child incrementing, value: 1
Parent incrementing, value: 2
Child incrementing, value: 3
Parent incrementing, value: 4
Child incrementing, value: 5
Parent incrementing, value: 6
Child incrementing, value: 7
Parent incrementing, value: 8
Child incrementing, value: 9
Parent incrementing, value: 10
Child incrementing, value: 11
Parent incrementing, value: 12
Child incrementing, value: 13
Parent incrementing, value: 14
Child incrementing, value: 15
Parent incrementing, value: 16
```