ITT440 INDIVIDUAL ASSIGNMENT

MUHAMMAD NUR ARIF BIN MUHAMMAD ASRI

2021603372

CS2554A

GITHUB LINK:

SOURCE CODE:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <signal.h>
#include <string.h>
#include <sys/types.h>

void handle_interrupt(int sig);
void handle_child(int pipe_fd);

int main() {
   int num_child;
   int pipe_fd[2];
   pid_t pid;
   char message[100];

   printf("Enter the number of child processes that you want to create: ");
   scanf("%d", &num_child);

   // Create the pipes for inter-process communication
   for (int i=0; i<num_child; i++) {
      if (pipe(pipe_fd) == -1) {
         perror("pipe");
         return 0;
      }
```

```
    }

    // Create the child processes
    for (int i=0; i<num_child; i++) {
        pid = fork();
        if (pid == -1) {
            perror("fork");
            return 0;
        }
        else if (pid == 0) {
            handle_child(pipe_fd[0]);
            exit(0);
        }
        else {
            // Parent process
            close(pipe_fd[0]); // Close the read end of the pipe
        }
    }

    // Set up the interrupt handler
    signal(SIGINT, handle_interrupt);

    // Loop to read messages from the user and pass them to the child processes
while (1) {
        printf("Enter a message to send to the child processes (or Ctrl+C to exit):\n");
        fgets(message, sizeof(message), stdin);
        if (strlen(message) > 1) { // ignore empty messages
                for (int j=0; j<num_child; j++) {
                        if (write(pipe_fd[1], message, strlen(message)) == -1) {
                        perror("write");
                        return 1;
```

```
                }
            }
        }
    }

    return 1;
}

// Handler function for SIGINT (Ctrl+C)
void handle_interrupt(int sig) {
    printf("\nInterrupt received. Exiting...\n");
    exit(0);
}

// Child process handler function
void handle_child(int pipe_fd) {
    char buffer[100];
    int num_read;

    while (1) {
        num_read = read(pipe_fd, buffer, sizeof(buffer));
        if (num_read == -1) {
            perror("read");
            exit(1);
        }
        else if (num_read == 0) {
            // End of file (pipe closed by parent)
            exit(0);
        }
        else {
            // Print the message received from the parent
```

```
        buffer[num_read] = '\0';
        printf("Child %d received message: %s", getpid(), buffer);
      }
    }
}
```

SAMPLE OUTPUT:

```
kaypaaa@kaype:~/ind_assign$ gcc -o ForkProg.out ForkProg.c
kaypaaa@kaype:~/ind_assign$ ./ForkProg.out
Enter the number of child processes that you want to create: 3
Enter a message to send to the child processes (or Ctrl+C to exit):
Enter a message to send to the child processes (or Ctrl+C to exit):
read: Bad file descriptor
read: Bad file descriptor
haii sayang
Enter a message to send to the child processes (or Ctrl+C to exit):
Child 1432 received message: haii sayang
haii sayang
haii sayang
jom kahwin
Enter a message to send to the child processes (or Ctrl+C to exit):
Child 1432 received message: jom kahwin
jom kahwin
jom kahwin
bye
Enter a message to send to the child processes (or Ctrl+C to exit):
Child 1432 received message: bye
bye
bye
^C
Interrupt received. Exiting...
kaypaaa@kaype:~/ind_assign$
```