CSE 344 HW2

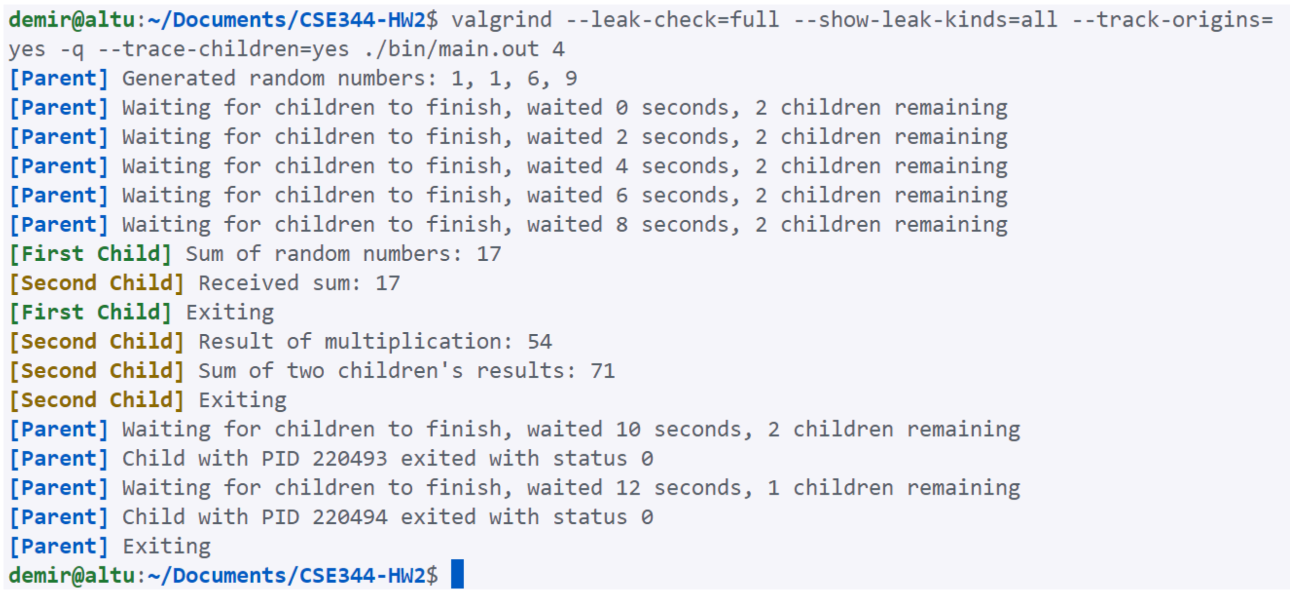
EMİRHAN ALTUNEL

200104004035

Code Outputs:

Expected Output:

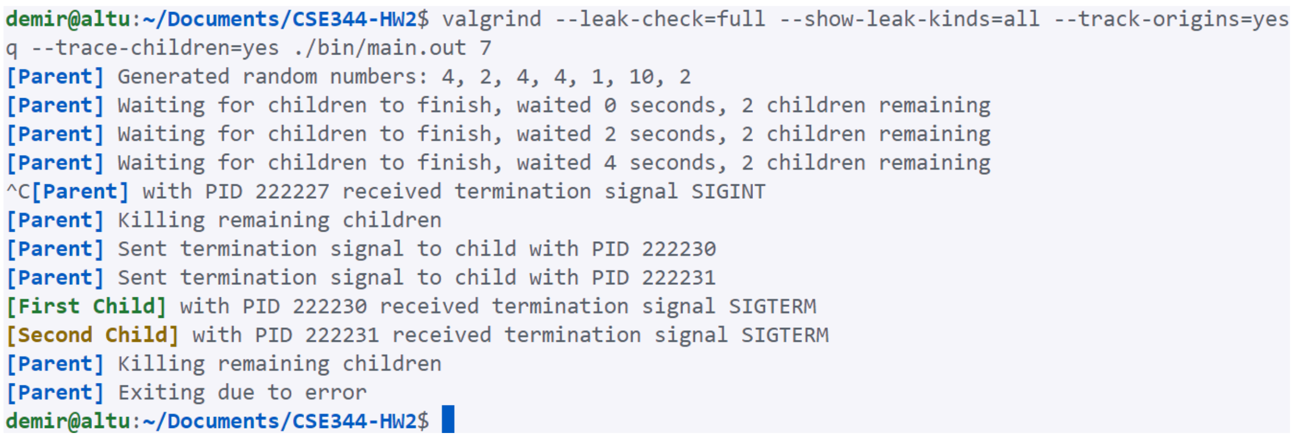
The parent should wait until its children finish.



```
demir@altu:~/Documents/CSE344-HW2$ valgrind --leak-check=full --show-leak-kinds=all --track-origins=
yes -q --trace-children=yes ./bin/main.out 4
[Parent] Generated random numbers: 1, 1, 6, 9
[Parent] Waiting for children to finish, waited 0 seconds, 2 children remaining
[Parent] Waiting for children to finish, waited 2 seconds, 2 children remaining
[Parent] Waiting for children to finish, waited 4 seconds, 2 children remaining
[Parent] Waiting for children to finish, waited 6 seconds, 2 children remaining
[Parent] Waiting for children to finish, waited 8 seconds, 2 children remaining
[First Child] Sum of random numbers: 17
[Second Child] Received sum: 17
[First Child] Exiting
[Second Child] Result of multiplication: 54
[Second Child] Sum of two children's results: 71
[Second Child] Exiting
[Parent] Waiting for children to finish, waited 10 seconds, 2 children remaining
[Parent] Child with PID 220493 exited with status 0
[Parent] Waiting for children to finish, waited 12 seconds, 1 children remaining
[Parent] Child with PID 220494 exited with status 0
[Parent] Exiting
demir@altu:~/Documents/CSE344-HW2$
```

While Parent waiting we sent CTRL + C or SIGTERM:

The parent send signal to its children to terminate them before exiting.



```
demir@altu:~/Documents/CSE344-HW2$ valgrind --leak-check=full --show-leak-kinds=all --track-origins=yes
q --trace-children=yes ./bin/main.out 7
[Parent] Generated random numbers: 4, 2, 4, 4, 1, 10, 2
[Parent] Waiting for children to finish, waited 0 seconds, 2 children remaining
[Parent] Waiting for children to finish, waited 2 seconds, 2 children remaining
[Parent] Waiting for children to finish, waited 4 seconds, 2 children remaining
^C[Parent] with PID 222227 received termination signal SIGINT
[Parent] Killing remaining children
[Parent] Sent termination signal to child with PID 222230
[Parent] Sent termination signal to child with PID 222231
[First Child] with PID 222230 received termination signal SIGTERM
[Second Child] with PID 222231 received termination signal SIGTERM
[Parent] Killing remaining children
[Parent] Exiting due to error
demir@altu:~/Documents/CSE344-HW2$
```

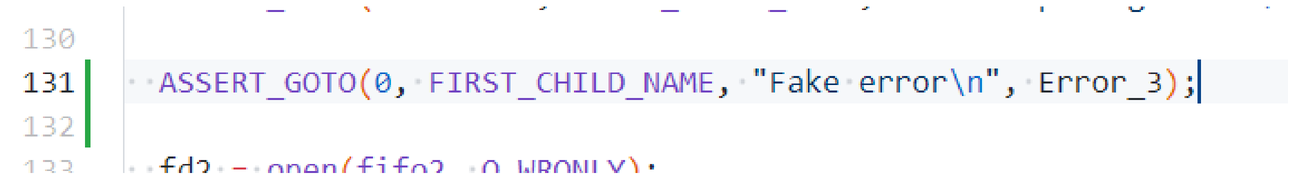
Error in child:

I added a fake error line in child 1 before its sleep. The parent catches that error and terminates the other child and exits

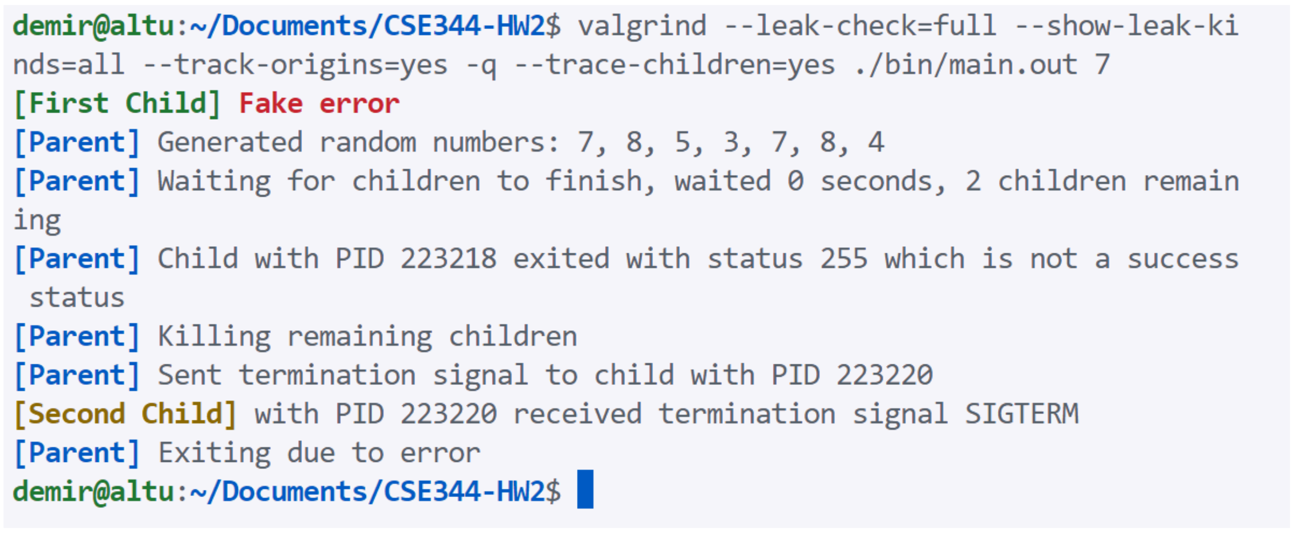
ASSERT_GOTO is a custom macro written by me.

ASSERT_GOTO(condition, sender, message, label);

If the condition is false it goto that label.



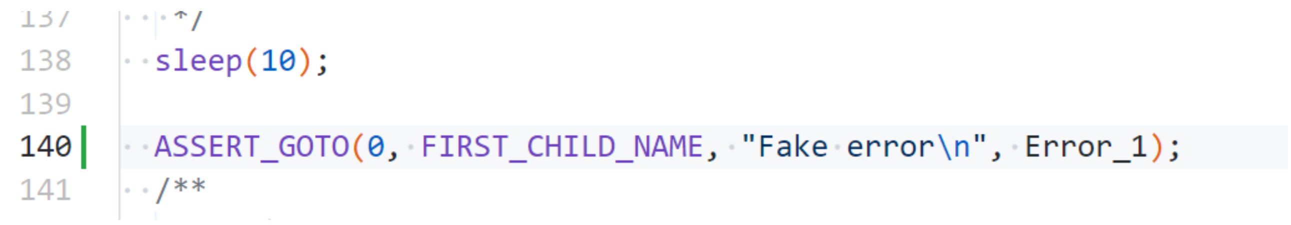
```
  ASSERT_GOTO(0, FIRST_CHILD_NAME, "Fake error\n", Error_3);

  fd2 = open(fifo2, O_WRONLY);
```



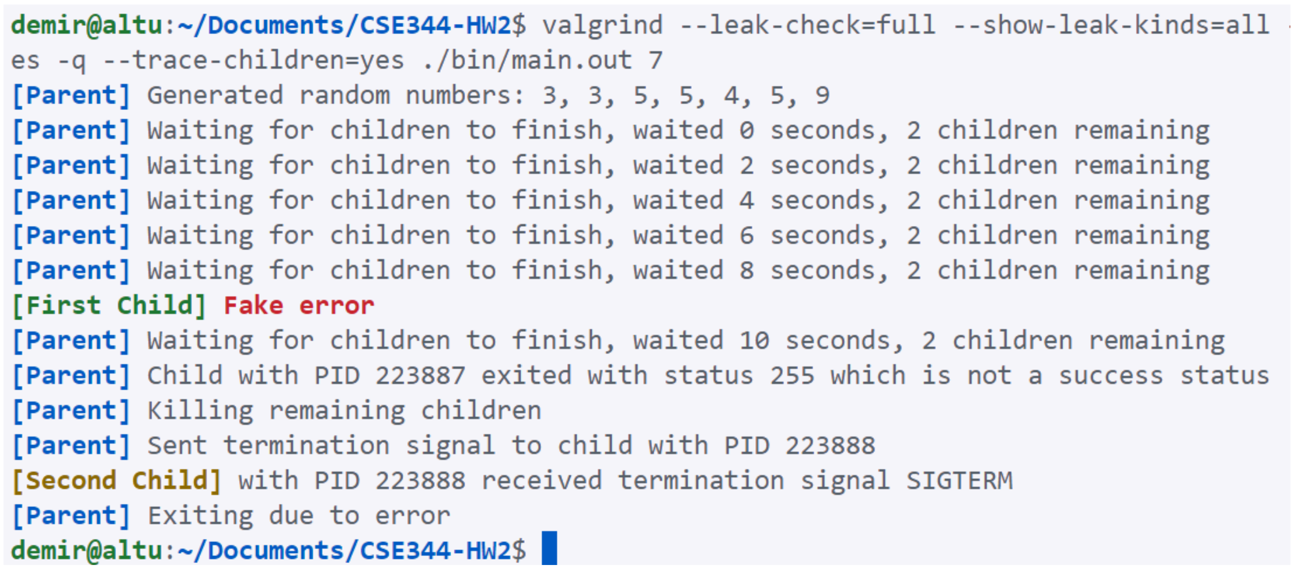
```
demir@altu:~/Documents/CSE344-HW2$ valgrind --leak-check=full --show-leak-ki
nds=all --track-origins=yes -q --trace-children=yes ./bin/main.out 7
[First Child] Fake error
[Parent] Generated random numbers: 7, 8, 5, 3, 7, 8, 4
[Parent] Waiting for children to finish, waited 0 seconds, 2 children remain
ing
[Parent] Child with PID 223218 exited with status 255 which is not a success
 status
[Parent] Killing remaining children
[Parent] Sent termination signal to child with PID 223220
[Second Child] with PID 223220 received termination signal SIGTERM
[Parent] Exiting due to error
demir@altu:~/Documents/CSE344-HW2$
```

Error in child:

I added a fake error line in child 1 after its sleep. The parent catches that error and terminates the other child and exits



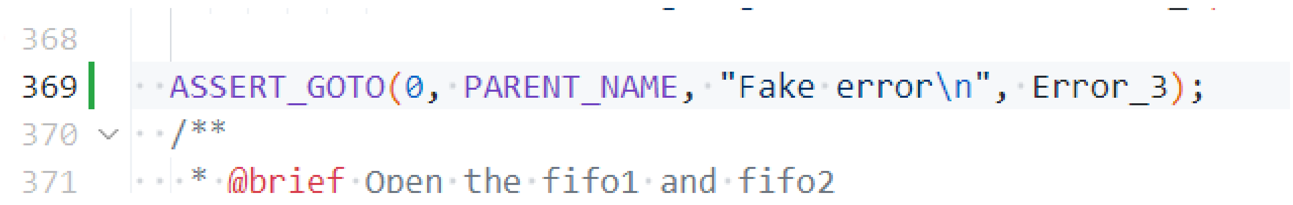
```
    */
  sleep(10);

  ASSERT_GOTO(0, FIRST_CHILD_NAME, "Fake error\n", Error_1);
  /**
```



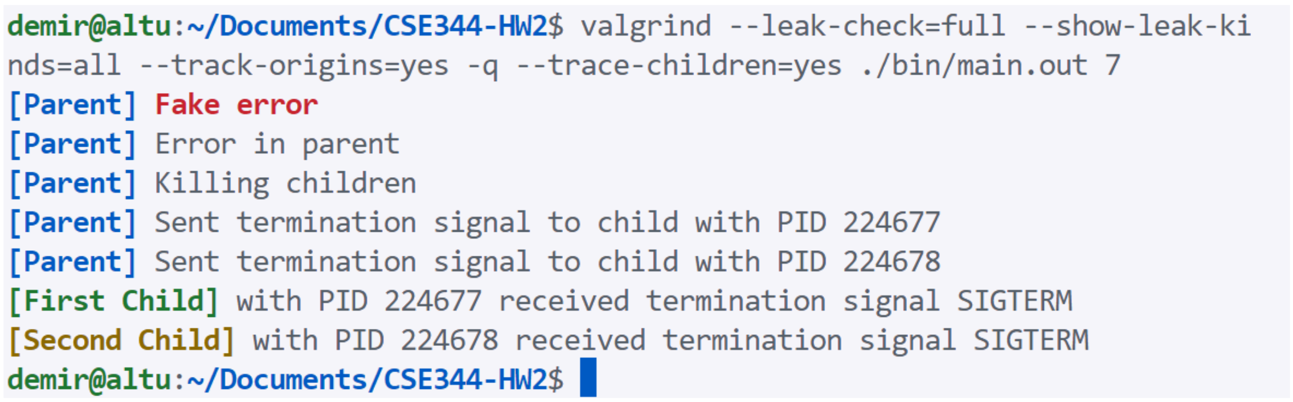
```
demir@altu:~/Documents/CSE344-HW2$ valgrind --leak-check=full --show-leak-kinds=all
es -q --trace-children=yes ./bin/main.out 7
[Parent] Generated random numbers: 3, 3, 5, 5, 4, 5, 9
[Parent] Waiting for children to finish, waited 0 seconds, 2 children remaining
[Parent] Waiting for children to finish, waited 2 seconds, 2 children remaining
[Parent] Waiting for children to finish, waited 4 seconds, 2 children remaining
[Parent] Waiting for children to finish, waited 6 seconds, 2 children remaining
[Parent] Waiting for children to finish, waited 8 seconds, 2 children remaining
[First Child] Fake error
[Parent] Waiting for children to finish, waited 10 seconds, 2 children remaining
[Parent] Child with PID 223887 exited with status 255 which is not a success status
[Parent] Killing remaining children
[Parent] Sent termination signal to child with PID 223888
[Second Child] with PID 223888 received termination signal SIGTERM
[Parent] Exiting due to error
demir@altu:~/Documents/CSE344-HW2$ 
```

Error in parent:

I added a fake error line in parent after. The parent should terminate all its children and exits.



```

  ASSERT_GOTO(0, PARENT_NAME, "Fake error\n", Error_3);
  /**
   * @brief Open the fifo1 and fifo2
```



```
demir@altu:~/Documents/CSE344-HW2$ valgrind --leak-check=full --show-leak-ki
nds=all --track-origins=yes -q --trace-children=yes ./bin/main.out 7
[Parent] Fake error
[Parent] Error in parent
[Parent] Killing children
[Parent] Sent termination signal to child with PID 224677
[Parent] Sent termination signal to child with PID 224678
[First Child] with PID 224677 received termination signal SIGTERM
[Second Child] with PID 224678 received termination signal SIGTERM
demir@altu:~/Documents/CSE344-HW2$ 
```

All of the errors run with valgrind and there is no leak. Valgrind messages don't show up because of the -q flag.

Implementations:

Parent’s Signal handler for SIGCHLD:

Process_safe_write is my function that is an alternative for fprintf.

SELF_EXIT is a special case, when a child gets a signal from others, it exits with that value. That helps for preventing multiple prints.

```c
void sigchld_handler(int signal) {
  if (signal != SIGCHLD) {
    return;
  }
  pid_t pid_child;
  int   status;
  int   return_value;

  while ((pid_child = waitpid(-1, &status, WNOHANG)) > 0) {
    return_value = WEXITSTATUS(status);
    child_count--;
    if (return_value == 0) {
      process_safe_write(1, "%s Child with PID %d exited with status %d\n",
                         PARENT_NAME, pid_child, return_value);
    } else {
      if (return_value != SELF_EXIT)
        process_safe_write(1,
                           "%s Child with PID %d exited with status %d "
                           "which is not a success status\n",
                           PARENT_NAME, pid_child, return_value);
      if (child_count > 0) {
        process_safe_write(1, "%s Killing remaining children\n", PARENT_NAME);
        kill_children();
        int status_remaining;
        while (waitpid(-1, &status_remaining, 0) > 0)
          ;
      }
      unlink_fifos();
      clear_all();
      process_safe_write(1, "%s Exiting due to error\n", PARENT_NAME);
      exit(0);
    }
  }
}
```

```c
/**
 * @brief Signal handler for SIGCHLD
 *
 */
struct sigaction sa = {0};
sa.sa_handler       = sigchld_handler;
ASSERT_GOTO(sigemptyset(&sa.sa_mask) != -1, PARENT_NAME,
            "Error initializing signal mask\n", Error_3);
ASSERT_GOTO(sigaction(SIGCHLD, &sa, NULL) != -1, PARENT_NAME,
            "Error setting signal handler\n", Error_3);
```

Writing from child 1:

```c
int sum = 0;
for (int i = 0; i < numberOfRandomNumbers; i++) sum += randomNumbers[i];
free(randomNumbers);
randomNumbers = NULL;
ASSERT_GOTO(write(fd2, &sum, sizeof(int)) != -1, FIRST_CHILD_NAME,
            "Error writing sum\n", Error_0);
```

Reading from child 2:

```c
int sum          = 0;
int return_value = -1;
do {
  return_value = read(fd2, &sum, sizeof(int));
  ASSERT_GOTO(return_value != -1, SECOND_CHILD_NAME, "Error reading sum\n",
              Error_0);
} while (return_value == 0);
close(fd2);
fd2 = -1;
process_safe_write(1, "%s Received sum: %d\n", SECOND_CHILD_NAME, sum);
```

There is a do while loop because if child 2 enters this section before child 1 opens fifo, read doesn’t work and doesn't block the child 2. So I added a soft block to that part.