CS5243 Advanced UNIX Programming
Assignment 8(4 pts)
Group 4

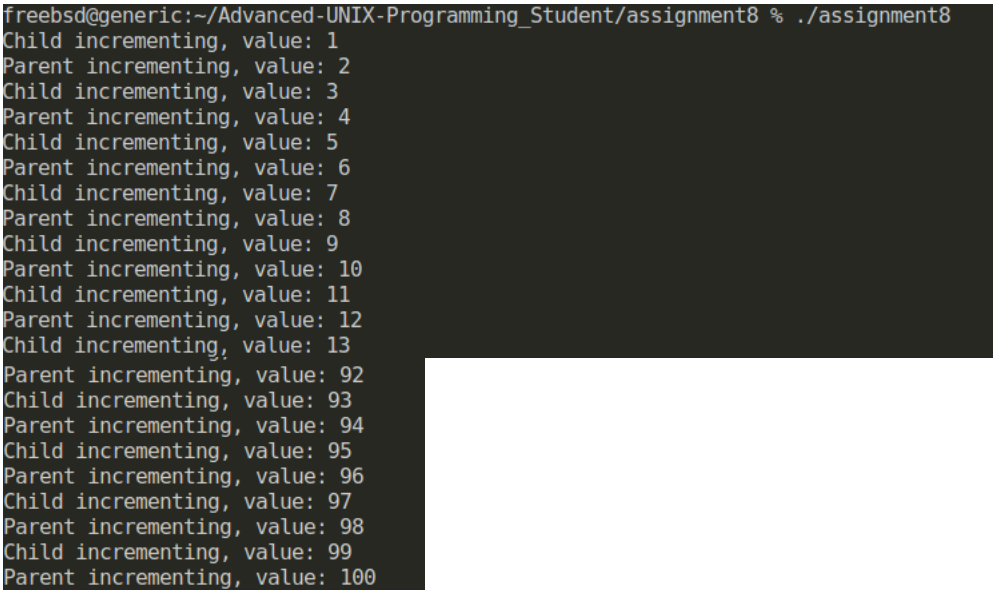
Screenshot of codes:

assignment8.c

```c
static int increment_counter(FILE *const file)
{
    /* TODO */
    char s[256];
    sprintf(s, "%d\n", counter);
    fwrite(s, strlen(s), 1, file);
    fflush(file);
    counter+=2;
    return counter - 2;
}

int main(void)
{
    /* TODO */

    // Open file
    FILE *fp;
    fp = fopen("counter.txt", "w+");

    // Create child process
    pid_t p = fork();
    if(p < 0){
        printf("Fork error\n");
        return -1;
    }
    if (p == 0){
        // Child process

        // Initialize signal handler for child
        if (signal(SIGUSR1, sig_usr) == SIG_ERR)
            perror("signal(SIGUSR1) error");

```

```
 99          // Set counter
100          counter = 1;
101
102          // Child first write file
103          TELL_WAIT();   // Signal Lock
104          printf("Child incrementing, value: %d\n", increment_counter(fp)); //Inc
105          TELL_PARENT(); //Signal Unlock
106
107          while(counter <= 99)
108          {
109              WAIT_PARENT();
110              TELL_WAIT();
111              printf("Child incrementing, value: %d\n", increment_counter(fp));
112              TELL_PARENT();
113
114          }
115      }else{
116          // Parent process
117
118          // Initialize signal handler for parent
119          if (signal(SIGUSR2, sig_usr) == SIG_ERR)
120              perror("signal(SIGUSR2) error");
121
122
123          // Set counter
124          counter = 2;
125
126
127          while(counter <= 100)
128          {
129              WAIT_CHILD();
130              TELL_WAIT();
131              printf("Parent incrementing, value: %d\n", increment_counter(fp));
132              TELL_CHILD(p);
133          }
134      }
135
136      fclose(fp);
137      return 0;
138
139  }
NORMAL  assignment8.c
:99
```

Screenshot of result:



Function *increment_counter,* is responsible for incrementing the global variable *counter* by 2.

For the main function, it forks a child process. The parent and child processes take turns incrementing the counter and writing the value to the file.

For child process:

- The child process initializes its signal handler for *SIGUSR1*.
- The child process uses the *TELL_WAIT* and *WAIT_PARENT* functions to synchronize with the parent.
- It increments the counter, writes the value to the file using *increment_counter*, and signals the parent process that it has finished.
- The child process then enters a loop, waiting for signals from the parent, incrementing the counter, and notifying the parent until the counter reaches 99.

For parent process:

- The parent process initializes its signal handler for *SIGUSR2*.
- It sets the initial value of the counter to 2.
- The parent process enters a loop, waiting for signals from the child, incrementing the counter, and notifying the child until the counter reaches 100.