

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

Screenshot of codes:

```
assignment8.c
67 static int increment_counter(FILE *const file)
68 {
69     /* TODO */
70     char s[256];
71     sprintf(s, "%d\n", counter);
72     fwrite(s, strlen(s), 1, file);
73     fflush(file);
74     counter+=2;
75     return counter - 2;
76 }
77
78 int main(void)
79 {
80     /* TODO */
81
82     // Open file
83     FILE *fp;
84     fp = fopen("counter.txt", "w+");
85
86     // Create child process
87     pid_t p = fork();
88     if(p < 0){
89         printf("Fork error\n");
90         return -1;
91     }
92     if (p == 0){
93         // Child process
94
95         // Initialize signal handler for child
96         if (signal(SIGUSR1, sig_usr) == SIG_ERR)
97             perror("signal(SIGUSR1) error");
98     }
```

```

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 }else{
115     // Parent process
116
117     // Initialize signal handler for parent
118     if (signal(SIGUSR2, sig_usr) == SIG_ERR)
119         perror("signal(SIGUSR2) error");
120
121
122     // Set counter
123     counter = 2;
124
125
126     while(counter <= 100)
127     {
128         WAIT_CHILD();
129         TELL_WAIT();
130         printf("Parent incrementing, value: %d\n", increment_counter(fp));
131         TELL_CHILD(p);
132     }
133 }
134
135 fclose(fp);
136 return 0;
137
138
139 }

```

NORMAL assignment8.c

:99

Screenshot of result:

```
freebsd@generic:~/Advanced-UNIX-Programming_Student/assignment8 % ./assignment8
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: 92
Child incrementing, value: 93
Parent incrementing, value: 94
Child incrementing, value: 95
Parent incrementing, value: 96
Child incrementing, value: 97
Parent incrementing, value: 98
Child incrementing, value: 99
Parent incrementing, value: 100
```

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.