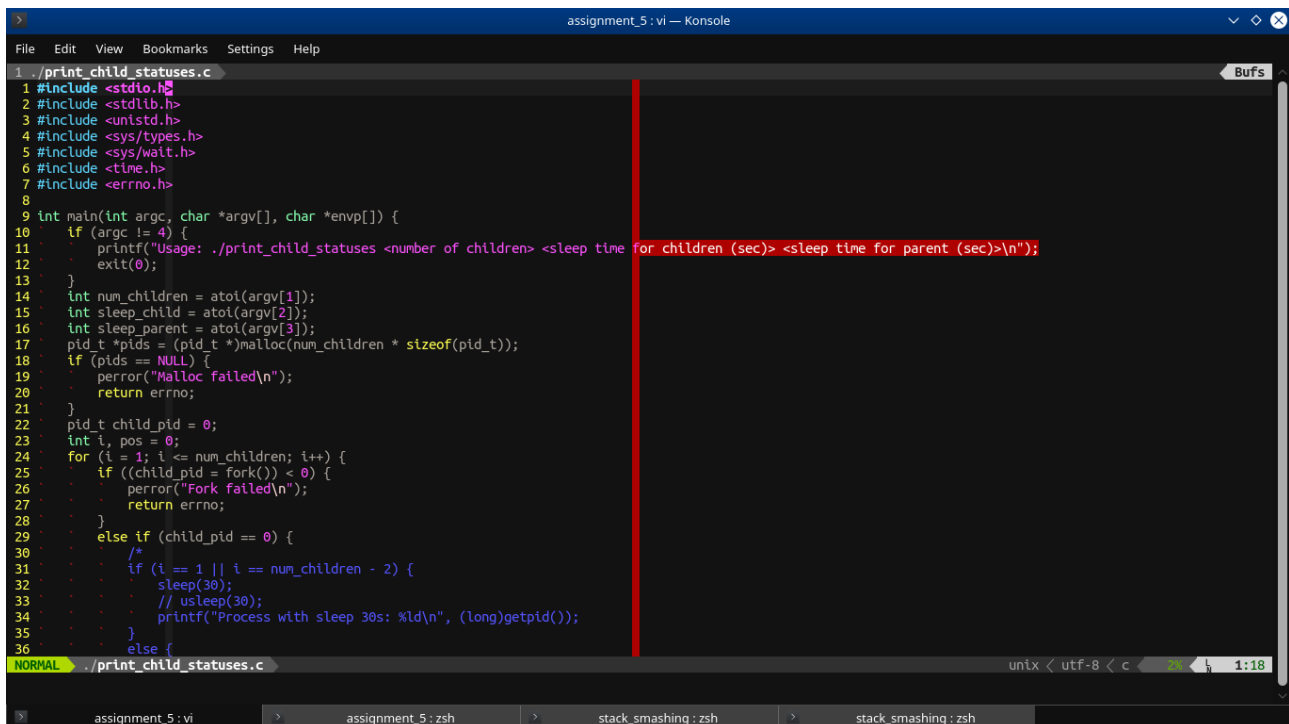


Assignment 5

Niramay Vaidya 111605075
Srishti Shelke 111603056

1. The parent starts as many child processes as to the value of its integer command line argument. The child processes simply sleep for the time specified by the argument, then exit. After starting all the children, the parent process does not wait for them immediately, but after a time specified by command line argument, checks the status of all terminated children, print the list of non terminated children and then terminates itself.

Code-



```
1 ./print_child_statuses.c
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <unistd.h>
5 #include <sys/types.h>
6 #include <sys/wait.h>
7 #include <time.h>
8 #include <errno.h>
9
10 int main(int argc, char *argv[], char *envp[]) {
11     if (argc != 4) {
12         printf("Usage: ./print_child_statuses <number of children> <sleep time for children (sec)> <sleep time for parent (sec)>\n");
13         exit(0);
14     }
15     int num_children = atoi(argv[1]);
16     int sleep_child = atoi(argv[2]);
17     int sleep_parent = atoi(argv[3]);
18     pid_t *pids = (pid_t *)malloc(num_children * sizeof(pid_t));
19     if (pids == NULL) {
20         perror("Malloc failed\n");
21         return errno;
22     }
23     pid_t child_pid = 0;
24     int i, pos = 0;
25     for (i = 1; i <= num_children; i++) {
26         if ((child_pid = fork()) < 0) {
27             perror("Fork failed\n");
28             return errno;
29         }
30         else if (child_pid == 0) {
31             /*
32              * if (i == 1 || i == num_children - 2) {
33              *     sleep(30);
34              *     // usleep(30);
35              *     printf("Process with sleep 30s: %ld\n", (long)getpid());
36              * }
37              */
38             else {
39                 sleep(sleep_child);
40                 exit(0);
41             }
42         }
43         pids[pos++] = child_pid;
44     }
45     sleep(sleep_parent);
46     for (i = 1; i <= num_children; i++) {
47         waitpid(pids[i-1], &status, 0);
48         if (WIFEXITED(status)) {
49             printf("Child %d exited with status %d\n", i, WEXITSTATUS(status));
50         }
51     }
52     printf("All children terminated\n");
53     return 0;
54 }
```

```
assignment_5: vi — Konsole
File Edit View Bookmarks Settings Help
1 ./print_child_statuses.c
37     sleep(sleep_child);
38     // usleep(sleep_child);
39 }
40 /*
41     sleep(sleep_child);
42     exit(i);
43 */
44 else {
45     pids[pos++] = child_pid;
46 }
47 }
48 sleep(sleep_parent);
49 // usleep(sleep_parent);
50 int status;
51 int ret;
52 for (i = 0; i < num_children; i++) {
53     if ((ret = waitpid(pids[i], &status, WNOHANG)) == -1) {
54         perror("Waitpid failed\n");
55         return errno;
56     }
57     else if (ret == 0) {
58         printf("Process with pid %ld not yet terminated\n", (long)pids[i]);
59     }
60     else {
61         printf("Process with pid %ld terminated with status %d\n", (long)pids[i], WIFEXITED(status) ? WEXITSTATUS(status) : 0);
62     }
63     free(pids);
64     return 0;
65 }
66 }

NORMAL ./print_child_statuses.c
unix < utf-8 < c 86% 57:28
assignment_5: vi assignment_5: zsh stack_smashing: zsh stack_smashing: zsh
```

Output-

```
initremay@itransoy: ~/Documents/sem7/AUP/assignment_5
$ ./print_child_statuses 5 2 2
Process with pid 23956 terminated with status 1
Process with pid 23957 terminated with status 2
Process with pid 23958 terminated with status 3
Process with pid 23959 terminated with status 4
Process with pid 23960 terminated with status 5
initremay@itransoy: ~/Documents/sem7/AUP/assignment_5
$ ./print_child_statuses 10 5 1
Process with pid 24008 not yet terminated
Process with pid 24009 not yet terminated
Process with pid 24010 not yet terminated
Process with pid 24011 not yet terminated
Process with pid 24012 not yet terminated
Process with pid 24013 not yet terminated
Process with pid 24014 not yet terminated
Process with pid 24015 not yet terminated
Process with pid 24016 not yet terminated
Process with pid 24017 not yet terminated
initremay@itransoy: ~/Documents/sem7/AUP/assignment_5
$ ./print_child_statuses 2 1 10
Process with pid 24065 terminated with status 1
Process with pid 24066 terminated with status 2
initremay@itransoy: ~/Documents/sem7/AUP/assignment_5
$ ./print_child_statuses 5 0 1
Process with pid 24116 terminated with status 1
Process with pid 24117 terminated with status 2
Process with pid 24118 terminated with status 3
Process with pid 24119 terminated with status 4
Process with pid 24120 terminated with status 5
initremay@itransoy: ~/Documents/sem7/AUP/assignment_5
```

(After uncommenting the block comment in the code and commenting out the call to sleep() below it)

```
initremay@itransoy: ~/Documents/sem7/AUP/assignment_5
$ ./print_child_statuses 25 1 2
Process with pid 24348 not yet terminated
Process with pid 24349 terminated with status 2
Process with pid 24350 terminated with status 3
Process with pid 24351 terminated with status 4
Process with pid 24352 terminated with status 5
Process with pid 24353 terminated with status 6
Process with pid 24354 terminated with status 7
Process with pid 24355 terminated with status 8
Process with pid 24356 terminated with status 9
Process with pid 24357 terminated with status 10
Process with pid 24358 terminated with status 11
Process with pid 24359 terminated with status 12
Process with pid 24360 terminated with status 13
Process with pid 24361 terminated with status 14
Process with pid 24362 terminated with status 15
Process with pid 24363 terminated with status 16
Process with pid 24364 terminated with status 17
Process with pid 24365 terminated with status 18
Process with pid 24366 terminated with status 19
Process with pid 24367 terminated with status 20
Process with pid 24368 terminated with status 21
Process with pid 24369 terminated with status 22
Process with pid 24370 not yet terminated
Process with pid 24371 terminated with status 24
Process with pid 24372 terminated with status 25
initremay@itransoy: ~/Documents/sem7/AUP/assignment_5
$ Process with sleep 30s: 24348
Process with sleep 30s: 24370
initremay@itransoy: ~/Documents/sem7/AUP/assignment_5
```

2. Assume that a process installs an exit handler and then forks a child. Does the child inherit the exit handler? Demonstrate.

Code-

```
assignment_5: vi — Konsole
File Edit View Bookmarks Settings Help
1 ./inherit_exit_handler.c BuFs
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <unistd.h>
5 #include <errno.h>
6
7 void exit_handler(void) {
8     printf("In the exit_handler\n");
9 }
10
11 int main(int argc, char *argv[], char *envp[]) {
12     if (atexit(exit_handler) != 0) {
13         perror("Atexit failed\n");
14         return errno;
15     }
16     pid_t pid = 0;
17     if ((pid = fork()) == -1) {
18         perror("Fork failed\n");
19         return errno;
20     }
21     else if (pid == 0) {
22         printf("Child is done\n");
23         exit(1);
24     }
25     else {
26         printf("Parent is done\n");
27         // printf("Parent is done\n");
28         return 0;
29     }
30 }
31
32 NORMAL ./inherit_exit_handler.c
33 "inherit_exit_handler.c" 29L, 514C written
34
35 assignment_5: vi assignment_5: zsh
```

Output-

```
niranmay@niranmay ~/Documents/sem7/AUP/assignment_5$ ./inherit_exit_handler
Parent is done
In the exit_handler
Child is done
In the exit_handler
niranmay@niranmay ~/Documents/sem7/AUP/assignment_5$
```