

Operating System Lab

Assignment 3

Arnav Samal
122CS0107

Q1.

Code:

```
1 #include <stdio.h>
2 #include <sys/types.h>
3 #include <unistd.h>
4
5 int main() {
6     fork();
7     fork();
8     printf("Hierarchy of 3 Processes\n");
9     return 0;
10 }
```

Output:

```
r-HP-Compaq-Elite-8300-SFF:~/Downloads/122CS0107/OperatingSystem/Lab_3$ gcc q1.c
nitr@nitr-HP-Compaq-Elite-8300-SFF:~/Downloads/122CS0107/OperatingSystem/Lab_3$ ./a.out
Hierarchy of 3 Processes
Hierarchy of 3 Processes
Hierarchy of 3 Processes
Hierarchy of 3 Processes
nitr@nitr-HP-Compaq-Elite-8300-SFF:~/Downloads/122CS0107/OperatingSystem/Lab_3$ |
```

Q2.

Code:

```

1 #include <stdio.h>
2 #include <unistd.h>
3
4 #define MAX_PROCESSES 7
5
6 int processTree[MAX_PROCESSES];
7 int pid;
8
9 void printTree(int index, int level) {
10     if (index >= MAX_PROCESSES) return;
11     printTree(2*index+2, level+1);
12     for (int i = 0; i < level; i++) printf(" ");
13     printf("Process %d\n", processTree[index]);
14     printTree(2*index+1, level+1);
15 }
16
17 int main() {
18     printf("Here I am just before first forking statement\n");
19     pid = fork();
20     processTree[0] = getpid();
21     if (pid == 0) {
22         printf("Here I am just after first forking statement\n");
23         pid = fork();
24         processTree[1] = getpid();
25         if (pid == 0) {
26             printf("Here I am just after second forking statement\n");
27             pid = fork();
28             processTree[3] = getpid();
29             if (pid != 0) {
30                 processTree[4] = getpid();
31             }
32         } else {
33             processTree[2] = getpid();
34             pid = fork();
35             processTree[5] = getpid();
36             if (pid != 0) {
37                 processTree[6] = getpid();
38             }
39         }
40     }
41     printf("Hello World from process %d!\n", getpid());
42     printTree(0, 0);
43     return 0;
44 }

```

Output:

```
nitr@nitr-HP-Compaq-Elite-8300-SFF: ~/Downloads/122CS0107/OperatingSystem/Lab_3
File Edit View Search Terminal Help
nitr@nitr-HP-Compaq-Elite-8300-SFF:~/Downloads/122CS0107/OperatingSystem/Lab_3$ ./a.out
Here I am just before first forking statement
Hello World from process 2965!
  Process 0
  Process 0
  Process 0
Process 2965
  Process 0
  Process 0
  Process 0
Here I am just after first forking statement
Hello World from process 2966!
  Process 2966
  Process 2966
  Process 2966
Process 2966
  Process 0
Here I am just after second forking statement
  Process 2966
  Process 0
Hello World from process 2968!
  Process 0
  Process 2966
  Process 2968
Process 2966
  Process 0
  Process 2966
  Process 0
Hello World from process 2967!
  Process 0
  Process 0
  Process 0
Process 2966
  Process 2967
  Process 2967
  Process 2967
Hello World from process 2969!
```

Q3.

Code:

```
1 #include <stdio.h>
2 #include <unistd.h>
3 #include <sys/types.h>
4 #include <sys/wait.h>
5
6 int main() {
7     pid_t pid1, pid2;
8
9     pid1 = fork();
10    if (pid1 == 0) {
11        for (int i = 1; i <= 10; i++) {
12            printf("P1: %d\n", i);
13            sleep(1);
14        }
15        return 0;
16    }
17
18    pid2 = fork();
19    if (pid2 == 0) {
20        for (char c = 'A'; c <= 'Z'; c++) {
21            printf("P2: %c\n", c);
22            sleep(1);
23        }
24        return 0;
25    }
26
27    printf("Parent process waiting for children to finish...\n");
28    waitpid(pid1, NULL, 0);
29    waitpid(pid2, NULL, 0);
30    printf("Parent process finished.\n");
31
32    return 0;
33 }
```

Output:

```
nitr@nitr-HP-Compaq-Elite-8300-SFF: ~/Downloads/122CS0107/OperatingSystem/Lab_3
File Edit View Search Terminal Help
Parent process waiting for children to finish...
P2: A
P1: 1
P2: B
P1: 2
P2: C
P1: 3
P1: 4
P2: D
P1: 5
P2: E
P2: F
P1: 6
P1: 7
P2: G
P1: 8
P2: H
P2: I
P1: 9
P2: J
P1: 10
P2: K
P2: L
P2: M
P2: N
P2: O
P2: P
P2: Q
P2: R
P2: S
P2: T
P2: U
P2: V
P2: W
P2: X
P2: Y
P2: Z
Parent process finished.
```

Q4.

Code(given):

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

```
int main(void)
{
    int i; for (i=0; i < 3; i++) {
        fork();
        printf("hello\n");
    }
    return 0;
}
```

Output:

```
nitr@nitr-HP-Compaq-Elite-8300-SFF: ~/Downloads/122CS0107/OperatingSystem/Lab_3$ gedit q4.c
nitr@nitr-HP-Compaq-Elite-8300-SFF: ~/Downloads/122CS0107/OperatingSystem/Lab_3$ gcc q4.c
nitr@nitr-HP-Compaq-Elite-8300-SFF: ~/Downloads/122CS0107/OperatingSystem/Lab_3$ ./a.out
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
nitr@nitr-HP-Compaq-Elite-8300-SFF: ~/Downloads/122CS0107/OperatingSystem/Lab_3$
```

Explanation:

When the code is initially run, it is first forked to make 2 processes child process and copy of parent process. These then print “hello”, and then it is forked again where 4 processes then print “hello”. This goes on one last time where it is forked for 3rd time and then 8 processes then print “hello”. In total $2+4+8 = 14$ “hello”s are printed.