

# Operating Systems

## HOMETASK#01



**23K-2001**

**BCS-4J**

Q1:

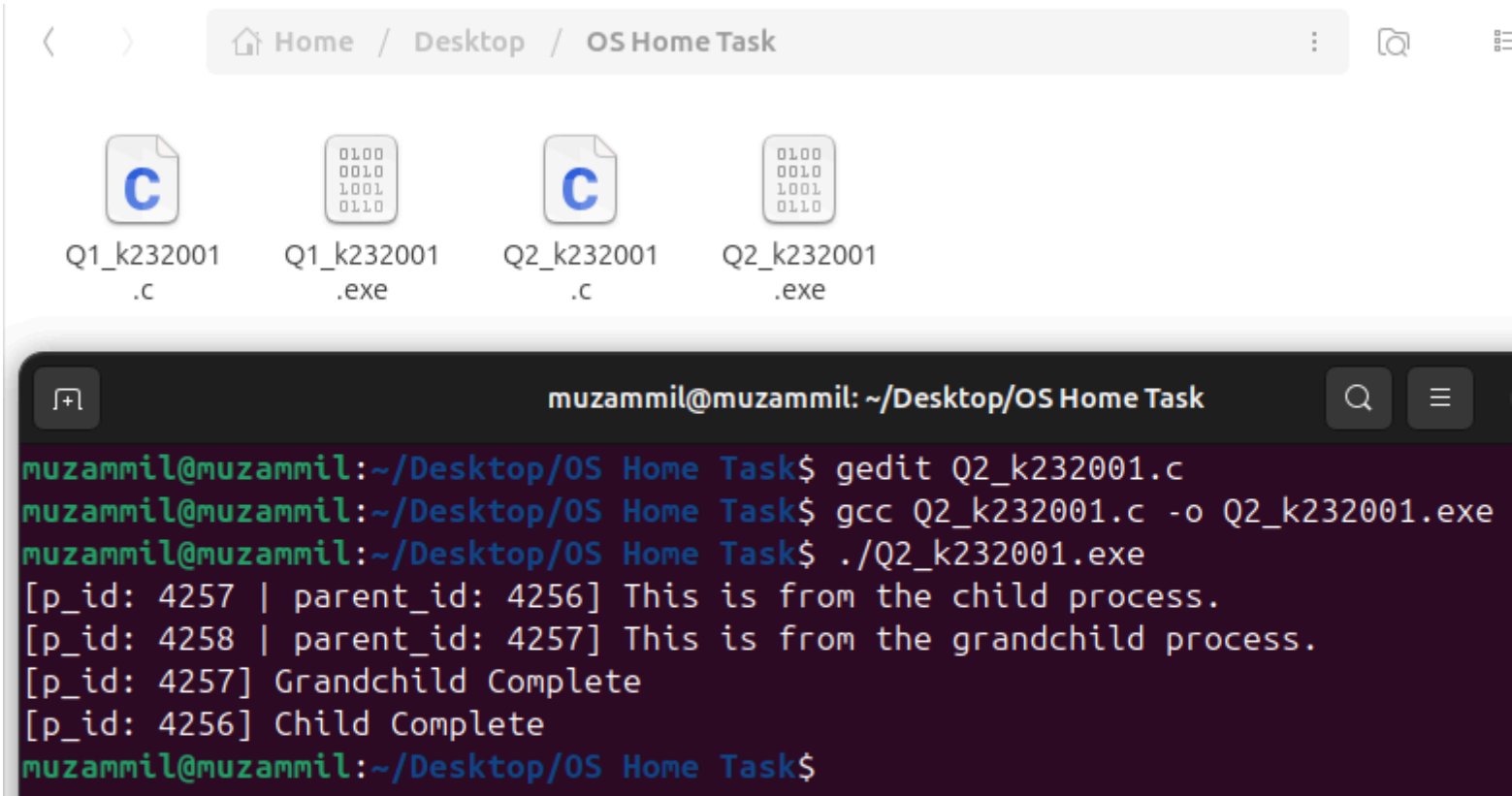
The screenshot displays a VirtualBox window titled 'Ubuntu 24.04.1 [Running] - Oracle VirtualBox'. The desktop environment is Ubuntu 24.04.1. A file manager window is open, showing the path 'Home / Desktop / OS Home Task' with two files: 'Q1\_k232001.c' and 'Q1\_k232001.exe'. A terminal window is open, showing the following commands and output:

```
 muzammil@muzammil: ~/Desktop/OS Home Task
 muzammil@muzammil:~/Desktop/OS Home Task$ gedit Q1_k232001.c
 muzammil@muzammil:~/Desktop/OS Home Task$ gcc Q1_k232001.c -o Q1_k232001.exe
 muzammil@muzammil:~/Desktop/OS Home Task$ ./Q1_k232001.exe
 Q1_k232001.c  Q1_k232001.exe
 Child Complete muzammil@muzammil:~/Desktop/OS Home Task$
```

A code editor window titled 'Q1\_k232001.c' is open, showing the following C code:

```
1 #include <sys/types.h>
2 #include <sys/wait.h>
3 #include <stdio.h>
4 #include <unistd.h>
5
6 int main()
7 {
8     pid_t pid;
9     /* fork a child process */
10    pid = fork();
11
12    if (pid < 0) { /* error occurred */
13        fprintf(stderr, "Fork Failed");
14        return 1;
15    }
16    else if (pid == 0) { /* child process */
17        execlp("/bin/ls", "ls", NULL);
18    }
19    else { /* parent process */
20        /* parent will wait for the child to complete */
21        wait(NULL);
22        printf("Child Complete");
23    }
24
25    return 0;
26 }
```

Q2:



```
Q2_k232001.c
~/Desktop/OS Home Task
Ln 38, Col 2
Q1_k232001.c
Q2_k232001.c
1 #include <sys/types.h>
2 #include <sys/wait.h>
3 #include <stdio.h>
4 #include <unistd.h>
5
6 int main()
7 {
8     pid_t pid1;
9     /* fork a child process */
10    pid1 = fork();
11    if (pid1 < 0) { /* error occurred */
12        fprintf(stderr, "Fork # 1 Failed");
13        return 1;
14    }
15    else if (pid1 == 0) { /* child process */
16        printf("[p_id: %d | parent_id: %d] This is from the child process.\n", getpid(), getppid());
17        pid_t pid2 = fork();
18        if (pid2 < 0) { /* error occurred */
19            fprintf(stderr, "Fork # 2 Failed");
20            return 1;
21        }
22        else if (pid2 == 0) { /* grandchild process */
23            printf("[p_id: %d | parent_id: %d] This is from the grandchild process.\n", getpid(), getppid());
24        }
25        else { /* parent process */
26            /* parent will wait for the child to complete */
27            wait(NULL);
28            printf("[p_id: %d] Grandchild Complete\n", getpid());
29        }
30    }
31
32    else { /* grandparent process */
33        /* parent will wait for the child to complete */
34        wait(NULL);
35        printf("[p_id: %d] Child Complete\n", getpid());
36    }
37    return 0;
38 }
```

```

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

int main()
{
    pid_t pid1;
    /* fork a child process */
    pid1 = fork();
    if (pid1 < 0) { /* error occurred */
        fprintf(stderr, "Fork # 1 Failed");
        return 1;
    }
    else if (pid1 == 0) { /* child process */
        printf("[p_id: %d | parent_id: %d] This is from the child process.\n", getpid(), getppid());
        pid_t pid2 = fork();
        if (pid2 < 0) { /* error occurred */
            fprintf(stderr, "Fork # 2 Failed");
            return 1;
        }
        else if (pid2 == 0) { /* grandchild process */
            printf("[p_id: %d | parent_id: %d] This is from the grandchild process.\n", getpid(),
getppid());
        }
        else { /* parent process */
            /* parent will wait for the child to complete */
            wait(NULL);
            printf("[p_id: %d] Grandchild Complete\n", getpid());
        }
    }

    else { /* grandparent process */
        /* parent will wait for the child to complete */
        wait(NULL);
        printf("[p_id: %d] Child Complete\n", getpid());
    }
    return 0;
}

```

Q3:

The screenshot displays a Linux desktop environment. At the top, a terminal window shows the command `gcc Q3_k232001.c -o Q3_k232001.exe` and the execution of the program `./Q3_k232001.exe`, which outputs a sequence of numbers: `0, 2, 4, 6, 8, 10, 12, 14, 16, 18 1, 3, 5, 7, 9, 11, 13, 15, 17, 19`. Below the terminal, a file manager window shows the contents of the `OS Home Task` directory, including files `Q1_k232001.c`, `Q1_k232001.exe`, `Q2_k232001.c`, `Q2_k232001.exe`, `Q3_k232001.c`, and `Q3_k232001.exe`. To the left, a code editor window displays the source code for `Q3_k232001.c`, which includes headers `<sys/types.h>`, `<sys/wait.h>`, `<stdio.h>`, and `<unistd.h>`. The code defines a `main` function that forks a child process. The child process prints odd numbers from 1 to 19, and the parent process prints even numbers from 0 to 18. The code is as follows:

```
1 #include <sys/types.h>
2 #include <sys/wait.h>
3 #include <stdio.h>
4 #include <unistd.h>
5
6 int main()
7 {
8     pid_t pid;
9     /* fork a child process */
10    pid = fork();
11
12    if (pid < 0) { /* error occurred */
13        fprintf(stderr, "Fork Failed");
14        return 1;
15    }
16    else if (pid == 0) { /* child process */
17        for(int i=1;i<20;i=i+2)
18            printf("%d ", i);
19    }
20    else { /* parent process */
21        for(int i=0;i<20;i=i+2)
22            printf("%d ", i);
23    }
24    printf("\n\n");
25    return 0;
26 }
```

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

int main()
{
    pid_t pid;
    /* fork a child process */
    pid = fork();

    if (pid < 0) { /* error occurred */
        fprintf(stderr, "Fork Failed");
        return 1;
    }
    else if (pid == 0) { /* child process */
        for(int i=1;i<20;i=i+2)
            printf("%d , ", i);
    }
    else { /* parent process */
        for(int i=0;i<20;i=i+2)
            printf("%d , ", i);
    }
    printf("\b\b");
    return 0;
}
```