

OS Lab 6



Session: 2021 – 2025

Submitted by:

Muhammad Yaqoob 2021-CS-118

Supervised by:

Ms. Abqa Javed

Department of Computer Science
University of Engineering and Technology
Lahore Pakistan

0.1 Lab Task

0.1.1 Question:

Implement date time server through message queues.

Hint:

The program creates a message queue then it forks.

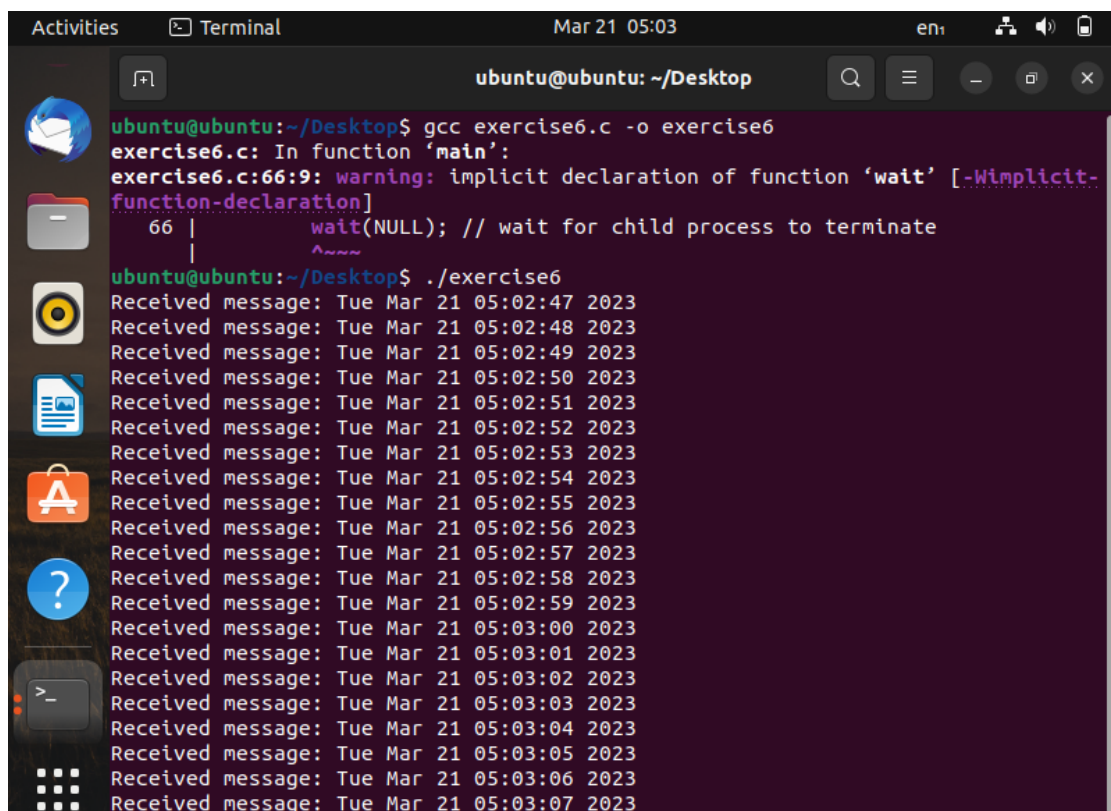
The child process is the consumer.

The parent process is the producer.

The producer continues to generate a date string and sends it in a message to the consumer.

The message type for a date message is one

0.1.2 Screenshot



```
ubuntu@ubuntu: ~/Desktop
ubuntu@ubuntu:~/Desktop$ gcc exercise6.c -o exercise6
exercise6.c: In function 'main':
exercise6.c:66:9: warning: implicit declaration of function 'wait' [-Wimplicit-function-declaration]
   66 |         wait(NULL); // wait for child process to terminate
       |         ^~~~~
ubuntu@ubuntu:~/Desktop$ ./exercise6
Received message: Tue Mar 21 05:02:47 2023
Received message: Tue Mar 21 05:02:48 2023
Received message: Tue Mar 21 05:02:49 2023
Received message: Tue Mar 21 05:02:50 2023
Received message: Tue Mar 21 05:02:51 2023
Received message: Tue Mar 21 05:02:52 2023
Received message: Tue Mar 21 05:02:53 2023
Received message: Tue Mar 21 05:02:54 2023
Received message: Tue Mar 21 05:02:55 2023
Received message: Tue Mar 21 05:02:56 2023
Received message: Tue Mar 21 05:02:57 2023
Received message: Tue Mar 21 05:02:58 2023
Received message: Tue Mar 21 05:02:59 2023
Received message: Tue Mar 21 05:03:00 2023
Received message: Tue Mar 21 05:03:01 2023
Received message: Tue Mar 21 05:03:02 2023
Received message: Tue Mar 21 05:03:03 2023
Received message: Tue Mar 21 05:03:04 2023
Received message: Tue Mar 21 05:03:05 2023
Received message: Tue Mar 21 05:03:06 2023
Received message: Tue Mar 21 05:03:07 2023
```

FIGURE 1: Screenshot

0.1.3 Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <unistd.h>
#include <sys/msg.h>
#include <sys/types.h>
#include <sys/ipc.h>
```

```
#define MSG_TYPE 1

struct message {
    long mtype; // message type
    char mtext[80]; // message text
};

int main() {
    key_t key = ftok("date-time-server", 'A');
    int msgid = msgget(key, IPC_CREAT | 0666);
    if (msgid == -1) {
        perror("msgget");
        exit(EXIT_FAILURE);
    }
    pid_t pid = fork();
    if (pid == -1) {
        perror("fork");
        exit(EXIT_FAILURE);
    } else if (pid == 0) { // child process: consumer
        while (1) {
            struct message msg;
            ssize_t n = msgrcv(msgid, &msg, sizeof(msg.mtext), MSG_TYPE, 0);
            if (n == -1) {
                perror("msgrcv");
                exit(EXIT_FAILURE);
            }
            if (strcmp(msg.mtext, "quit") == 0) {
                printf("Terminating child process\n");
                break;
            }
            printf("Received message: %s\n", msg.mtext);
        }
    } else { // parent process: producer
        while (1) {
            time_t t = time(NULL);
            struct tm *tm = localtime(&t);
            char buf[80];
            strftime(buf, sizeof(buf), "%c", tm);
            struct message msg;
            msg.mtype = MSG_TYPE;
            strncpy(msg.mtext, buf, sizeof(msg.mtext));
            int ret = msgsnd(msgid, &msg, sizeof(msg.mtext), 0);
            if (ret == -1) {
                perror("msgsnd");
                exit(EXIT_FAILURE);
            }
            sleep(1);
        }
        struct message msg;
        msg.mtype = MSG_TYPE;
        strcpy(msg.mtext, "quit");
        int ret = msgsnd(msgid, &msg, sizeof(msg.mtext), 0);
        if (ret == -1) {
            perror("msgsnd");
        }
    }
}
```

```
        exit(EXIT_FAILURE);
    }
    wait(NULL); // wait for child process to terminate
    msgctl(msgid, IPC_RMID, NULL); // delete message queue
}
return 0;
}
```
