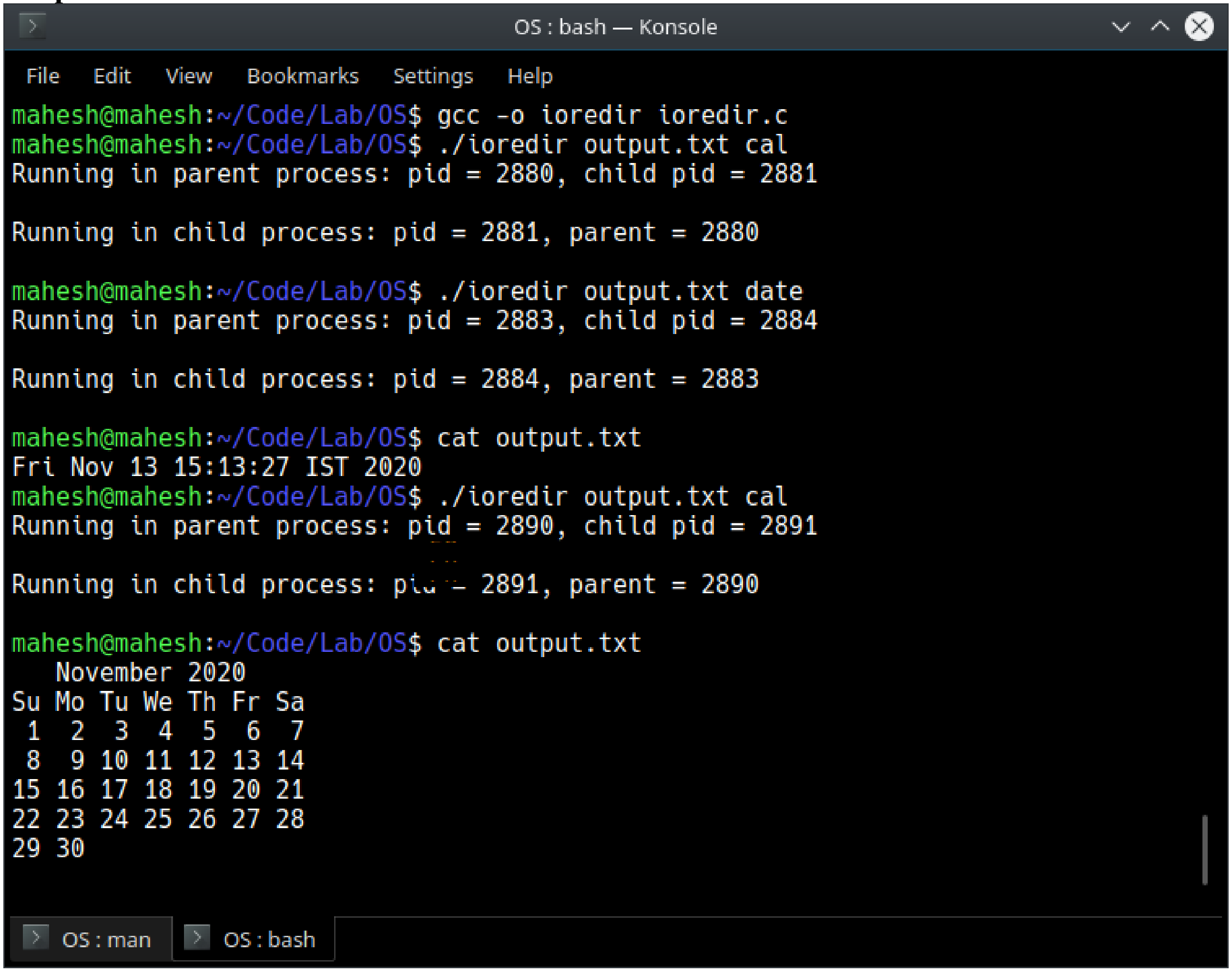
Program 2.

Write a program that demonstrates fork, exec, wait and getpid system calls.

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

// Program 2: Demonstrate use of exec, fork, wait
// This program writes the output of any program to file specified
by user
// Using the same strategy used by I/O redirection in unix shells

// The principle is, file descripters survive through
// invocations of execve syscalls

// This program prints PID of parent and child from both processes
// using getpid and getppid calls

int main(int argc, char *argv[]) {
     int fd, forkret;
     if(argc < 3) {
          fprintf(stderr, "Need at least 2 arguments\n");
          return 1;
     }

     fd = open(argv[1], O_WRONLY | O_TRUNC | O_CREAT, S_IRUSR |
S_IWUSR);
     if (fd < 0) {
          perror("Failed to open file");
          return 1;
     }

     // File descriptors survive across execve() calls
     // Using dup2 to clone fd to stdout

     dup2(fd, 1); // 1 is stdout

     forkret = fork();
     if (forkret < 0) {
          perror("Fork failed");
          return 1;
     } else if (forkret == 0) {
          fprintf(stderr,
               "Running in child process: pid = %d, parent = %d\n\
n",
               (int)getpid(),
               (int)getppid()
          );
          // child process, execute the command
          execvp(argv[2], &argv[2]);
```

```
            perror("Exec failed");
        } else {
            fprintf(stderr,
                "Running in parent process: pid = %d, child pid =
%d\n\n",
                getpid(),
                forkret
            );
            wait(NULL);
        }
        return 0;
}
```

Output:



```
mahesh@mahesh:~/Code/Lab/OS$ gcc -o ioredir ioredir.c
mahesh@mahesh:~/Code/Lab/OS$ ./ioredir output.txt cal
Running in parent process: pid = 2880, child pid = 2881

Running in child process: pid = 2881, parent = 2880

mahesh@mahesh:~/Code/Lab/OS$ ./ioredir output.txt date
Running in parent process: pid = 2883, child pid = 2884

Running in child process: pid = 2884, parent = 2883

mahesh@mahesh:~/Code/Lab/OS$ cat output.txt
Fri Nov 13 15:13:27 IST 2020
mahesh@mahesh:~/Code/Lab/OS$ ./ioredir output.txt cal
Running in parent process: pid = 2890, child pid = 2891

Running in child process: pid = 2891, parent = 2890

mahesh@mahesh:~/Code/Lab/OS$ cat output.txt
   November 2020
Su Mo Tu We Th Fr Sa
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30
```