Assignment 7(a): Signal

Implementing signals for Inter-process communication in a C program:

a. Signals are limited forms of IPC
b. Signals are asynchronous notifications sent to a process/thread.

C function used to create unnamed pipes:

```
        #include <signal.h>
        typedef void (*signalhandler_t)(int);
        signalhandler_t signal(int signal_num, signalhandler_t handler);
```

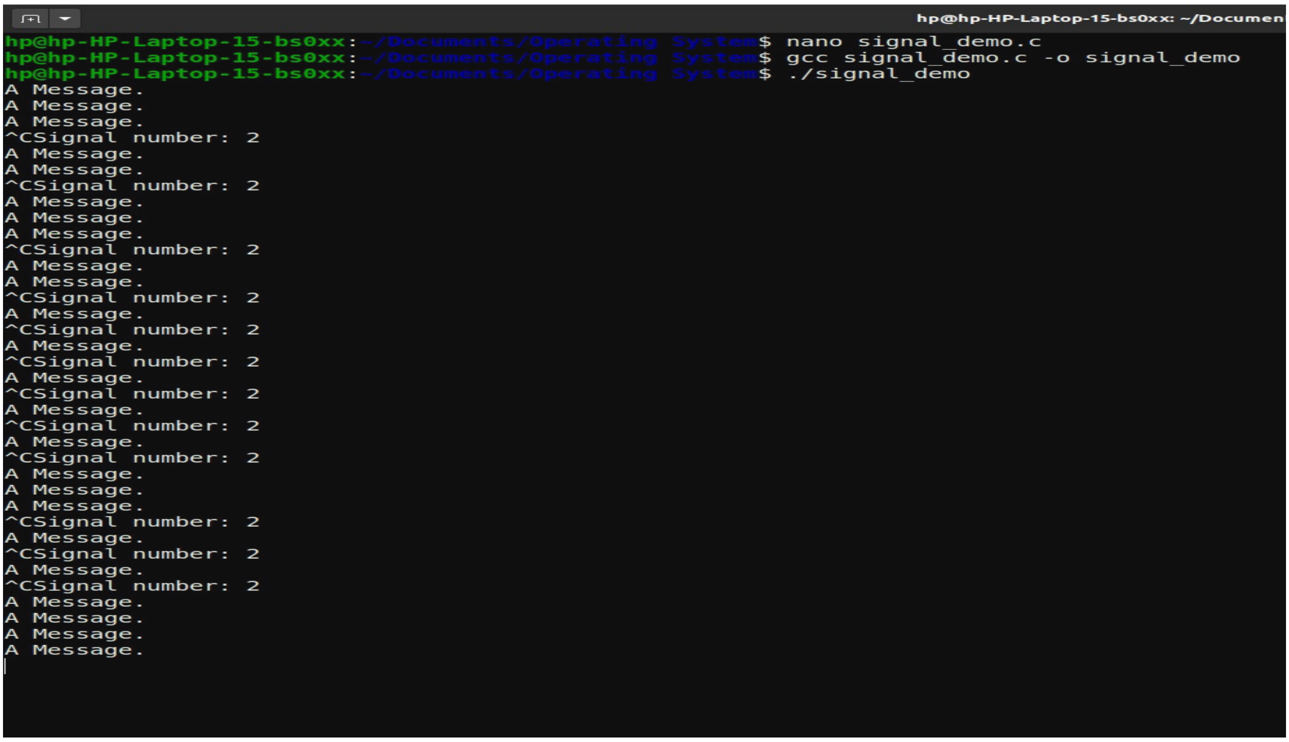
Sample code:

```
// C program to illustrate

void fun1(int signum) {
        printf("Signal number: %d\n", signum); // whatever you want to do to handle the signal
}

int main() {
        signal(SIGINT, fun1);
        while (1) {
                printf("A Message.\n");
                sleep(1); // to sync the output
        }
}
```

Output:



```
hp@hp-HP-Laptop-15-bs0xx:~/Documents/Operating System$ nano signal_demo.c
hp@hp-HP-Laptop-15-bs0xx:~/Documents/Operating System$ gcc signal_demo.c -o signal_demo
hp@hp-HP-Laptop-15-bs0xx:~/Documents/Operating System$ ./signal_demo
A Message.
A Message.
A Message.
^CSignal number: 2
A Message.
A Message.
^CSignal number: 2
A Message.
A Message.
A Message.
^CSignal number: 2
A Message.
A Message.
^CSignal number: 2
A Message.
^CSignal number: 2
A Message.
^CSignal number: 2
A Message.
^CSignal number: 2
A Message.
^CSignal number: 2
A Message.
^CSignal number: 2
A Message.
A Message.
A Message.
^CSignal number: 2
A Message.
^CSignal number: 2
A Message.
^CSignal number: 2
A Message.
A Message.
A Message.
A Message.
```

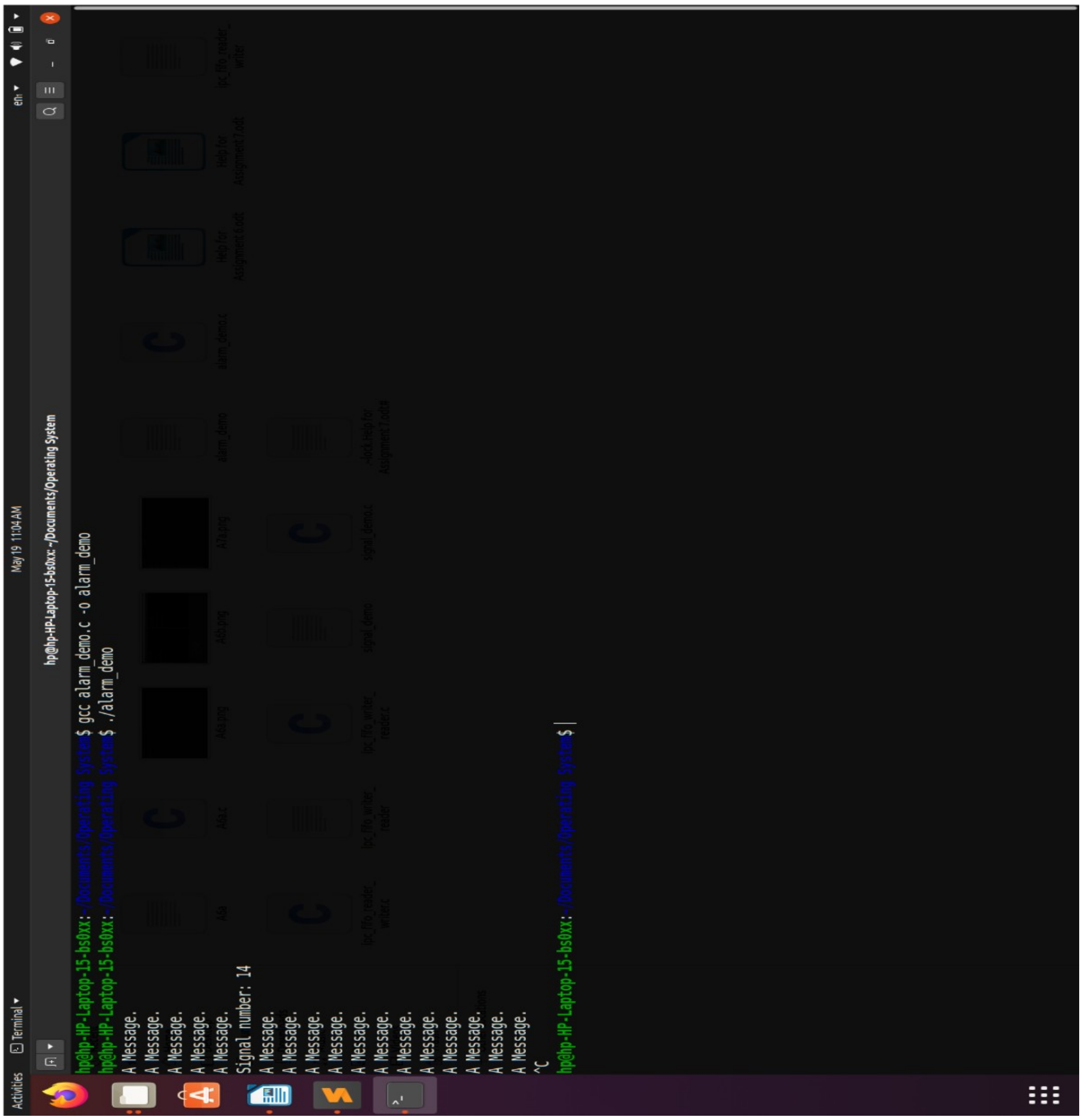
Assignment 7(b): Alarm Signal

To send Alarm signal from one process/thread to another use "kill" system call.

Sample code:

```
void fun1(int signum) {
// whatever you want to do to handle the signal
        printf("Signal number: %d\n", signum);
}
int main() {
        int pid = fork(); // child thread create
        if(pid == 0) { // child
                sleep(5);
                kill(getppid(), SIGALRM); // sending signal to parent
                exit(0);
        } else {
                signal(SIGALRM, fun1);
        while (1) {
            printf("A Message.\n");
            sleep(1); // to sync the output
        }
    }
}
```

Output:



```
hp@hp-HP-Laptop-15-bs0xx:~/Documents/Operating System$ gcc alarm_demo.c -o alarm_demo
hp@hp-HP-Laptop-15-bs0xx:~/Documents/Operating System$ ./alarm_demo
A Message.
A Message.
A Message.
A Message.
A Message.
Signal number: 14
A Message.
A Message.
A Message.
A Message.
A Message.
A Message.
A Message.
A Message.
A Message.
A Message.
A Message.
A Message.
^C
hp@hp-HP-Laptop-15-bs0xx:~/Documents/Operating System$ 
```

Assignment 7(c): Alarm Signal

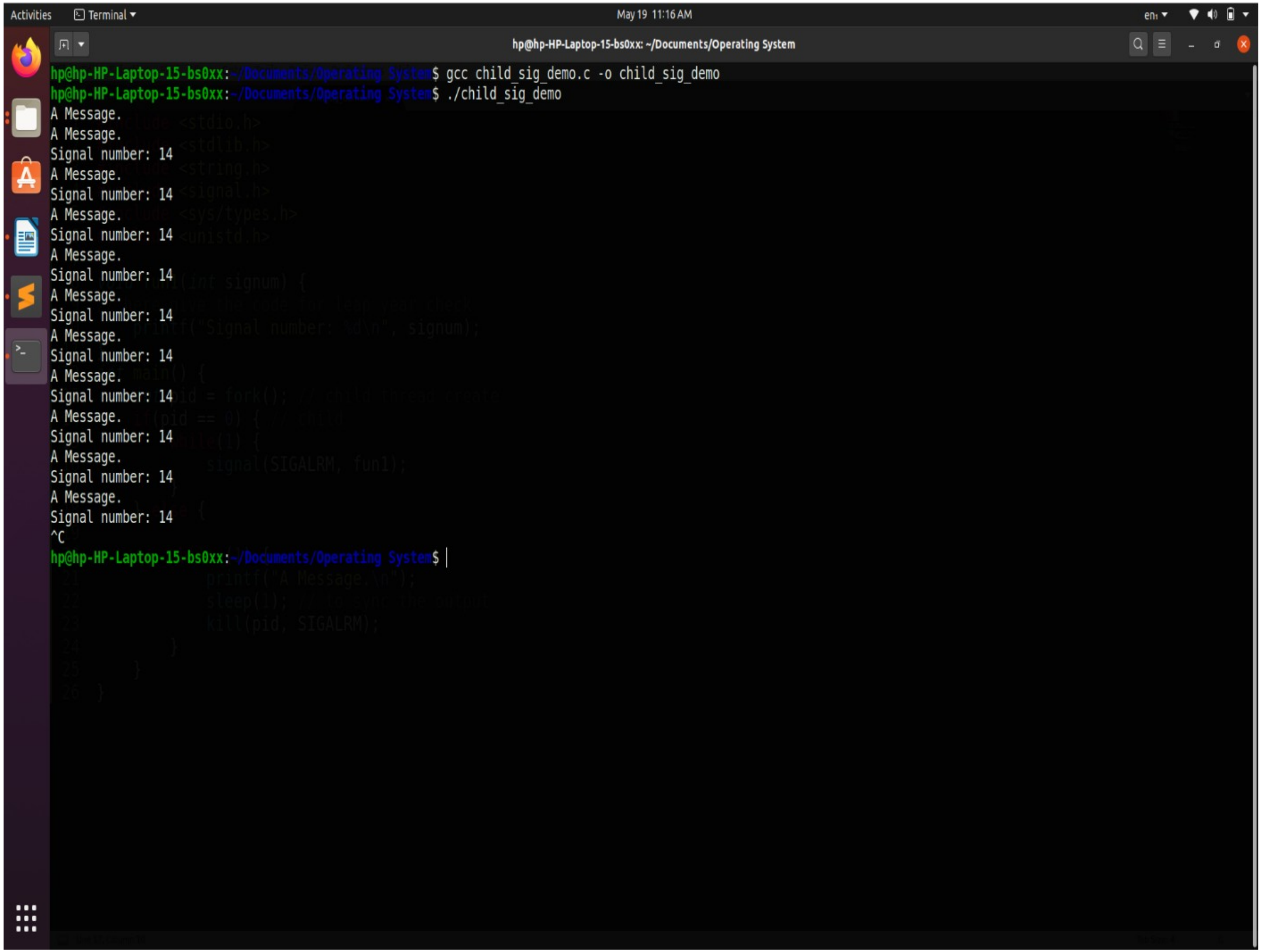
To send Alarm signal from one process/thread to another use “kill” system call.

Sample code:

```
void fun1(int signum) {
// here give the code for leap year check
        printf("Signal number: %d\n", signum);
}
int main() {
        int pid = fork(); // child thread create
        if(pid == 0) { // child
                while(1) {
                        signal(SIGALRM, fun1); // handling alarm
                }
        } else {

                while (1) {
                        printf("A Message.\n");
                        sleep(5); // to sync the output
                        kill(pid, SIGALRM);
                }
        }
}
```

Output:



```
hp@hp-HP-Laptop-15-bs0xx:~/Documents/Operating System$ gcc child_sig_demo.c -o child_sig_demo
hp@hp-HP-Laptop-15-bs0xx:~/Documents/Operating System$ ./child_sig_demo
A Message.
A Message.
Signal number: 14
A Message.
Signal number: 14
A Message.
Signal number: 14
A Message.
Signal number: 14
A Message.
Signal number: 14
A Message.
Signal number: 14
A Message.
Signal number: 14
A Message.
Signal number: 14
A Message.
Signal number: 14
A Message.
Signal number: 14
^C
hp@hp-HP-Laptop-15-bs0xx:~/Documents/Operating System$ 
```