Exercise problems on signal system call

1. **Basic Signal Handling**
   Write a C program that uses the `signal` system call to catch the `SIGINT` signal (Ctrl+C) and prints a custom message when the signal is received.

2. **Signal Handling with Fork**
   Extend the previous program to include a child process created using `fork()`. Ensure that both the parent and child processes have signal handlers for `SIGINT` and can handle the signal independently.

3. **Ignoring a Signal**
   Write a program that sets up a signal handler for `SIGTERM` to simply print a message and then ignores the `SIGTERM` signal for a certain period. After the period expires, the program should allow `SIGTERM` to terminate it.

4. **Signal Masking**
   Create a program that demonstrates signal masking using `sigprocmask`. The program should block the `SIGUSR1` signal, send this signal to itself, and observe that it remains blocked. After a certain point, unblock the signal and allow it to be caught.

5. **Signal Queue**
   Write a C program that uses signals to implement a simple communication mechanism between two processes. One process should send a custom signal (e.g., `SIGUSR1`) to the other process, and the receiving process should print a message upon receiving the signal.

6. **Handling Multiple Signals**
   Create a program that handles multiple signals (`SIGINT`, `SIGTERM`, `SIGUSR1`, and `SIGUSR2`) with custom handlers. The program should print different messages depending on which signal is received.

7. **Signal Interruption**
   Write a program that performs a time-consuming operation (e.g., a loop) and can be interrupted by the `SIGINT` signal. When `SIGINT` is received, the program should print a message and gracefully exit.

8. **Signal Error Handling**
   Develop a program that demonstrates error handling related to signal operations. For instance, try to send a signal to a non-existent process and handle the error appropriately.

9. **Handling Real-Time Signals**
   Create a program that uses real-time signals (e.g., `SIGRTMIN+1` and `SIGRTMIN+2`) and custom signal handlers to perform a specific task. Test the program by sending these real-time signals to it.

10. **Signal Synchronization**
    Write a program that uses signals to synchronize two processes. One process should wait for a signal from the other process before proceeding. Demonstrate how to achieve this synchronization effectively.

Remember to include error checking in your code, especially when using system calls like `signal` and `sigprocmask`.