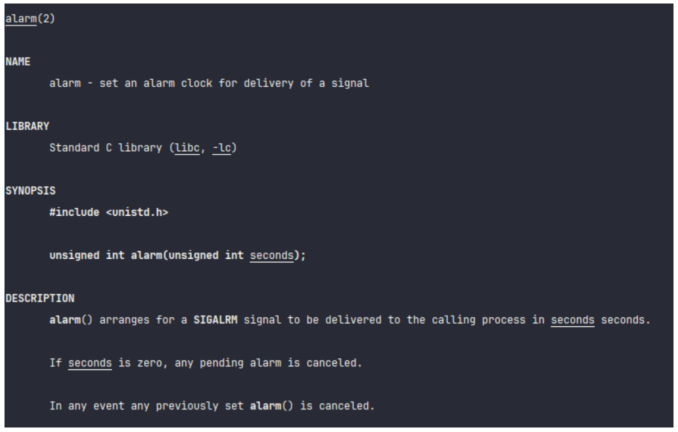
Exercise

- **Make queued signal**
  - **Sender** is the parent process and
  - **Receiver** is the child process
  - The program get the number of signals to send by argument (**argv**)
  - Use **SIGUSR1, SIGALRM, SIGINT** signals



```
alarm(2)

NAME
       alarm - set an alarm clock for delivery of a signal

LIBRARY
       Standard C library (libc, -lc)

SYNOPSIS
       #include <unistd.h>

       unsigned int alarm(unsigned int seconds);

DESCRIPTION
       alarm() arranges for a SIGALRM signal to be delivered to the calling process in seconds seconds.

       If seconds is zero, any pending alarm is canceled.

       In any event any previously set alarm() is canceled.
```

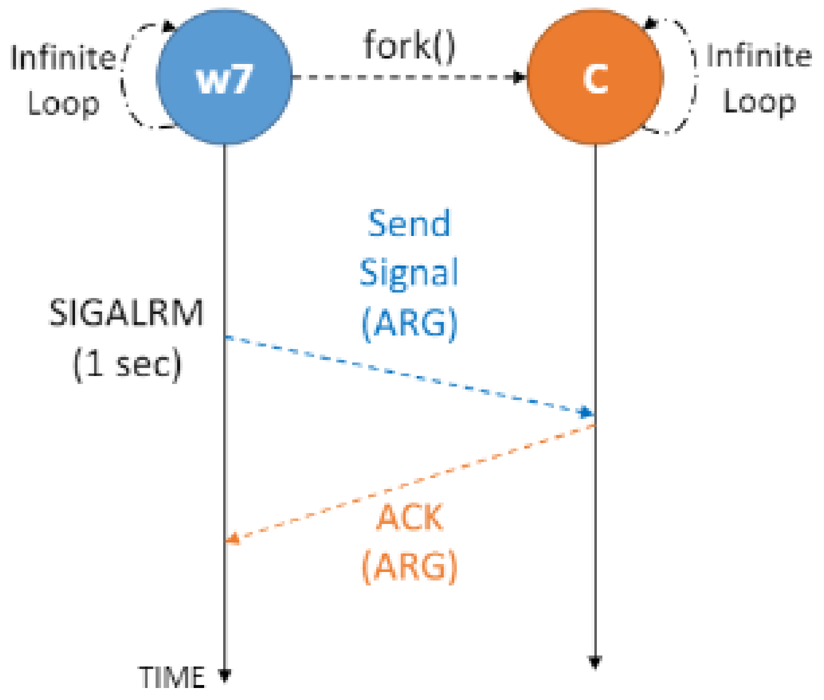
Exercise

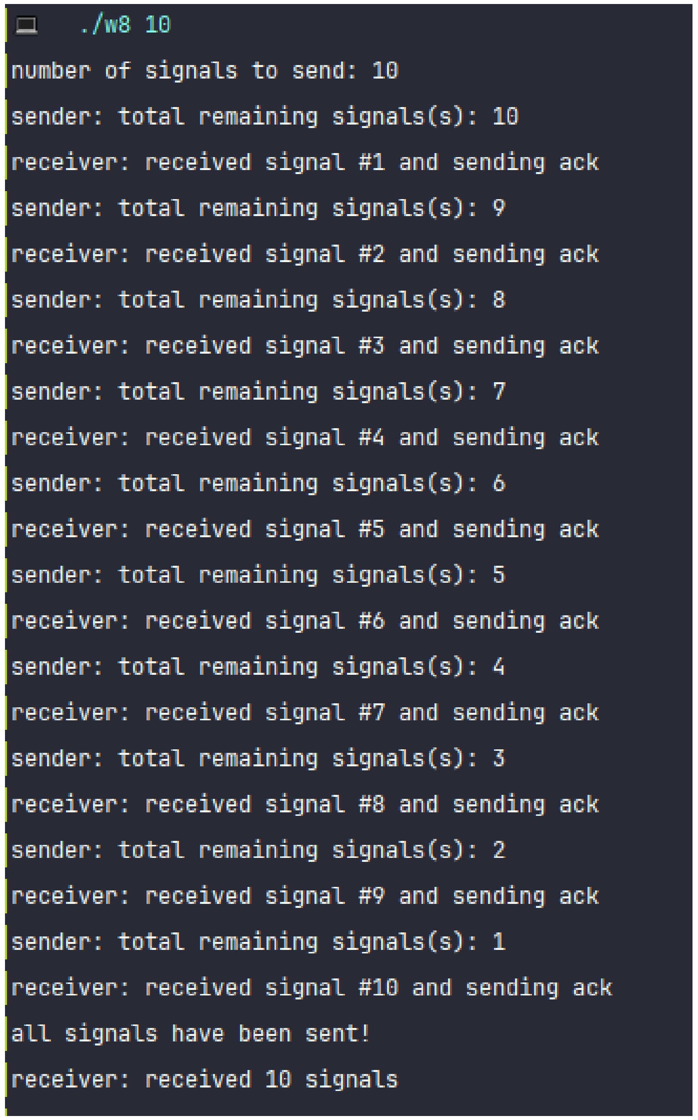
- **Sender**
  - Sender **sends a signal** to the receiver and **receives an acknowledgment** from the receiver
  - Sender checks the number of sent signals and received acknowledgments
  - If Sender doesn't receive all acks of sending signal, send back the remaining signals after 1 second
    - » **Do not use** *wait()* or *sleep*(); use ***alarm()*** function instead
    - » ***alarm()*** is already in the skeleton code
  - If the number of signals sent and the number of acks received are the same, send **SIGINT** to receiver and terminate itself

Exercise

- **Receiver**
  - Receiver **receives a signal** from the sender and **sends an acknowledgment** to the sender
  - If receiver receive **SIGINT**, Receiver prints how many signal it received and then terminates
- **Hint**
  - The sender and receiver do not share the values in the variable. They have their own copy of the variable.

Exercise running example





```
 ./w8 10
number of signals to send: 10
sender: total remaining signals(s): 10
receiver: received signal #1 and sending ack
sender: total remaining signals(s): 9
receiver: received signal #2 and sending ack
sender: total remaining signals(s): 8
receiver: received signal #3 and sending ack
sender: total remaining signals(s): 7
receiver: received signal #4 and sending ack
sender: total remaining signals(s): 6
receiver: received signal #5 and sending ack
sender: total remaining signals(s): 5
receiver: received signal #6 and sending ack
sender: total remaining signals(s): 4
receiver: received signal #7 and sending ack
sender: total remaining signals(s): 3
receiver: received signal #8 and sending ack
sender: total remaining signals(s): 2
receiver: received signal #9 and sending ack
sender: total remaining signals(s): 1
receiver: received signal #10 and sending ack
all signals have been sent!
receiver: received 10 signals
```

Exercise submission

- **Submit your source code and Makefile**
  - via **iCampus**
  - Makefile should generate ./w7 executable
  - Bundle *source code* and *Makefile* with tar command
    - » *tar.gz* format

    ```
    $ tar cvzf [student_id].tar.gz week7
    ```
  - We'll grade your submission with **make**
    - » If compilation fails, your points for this exercise will be zero