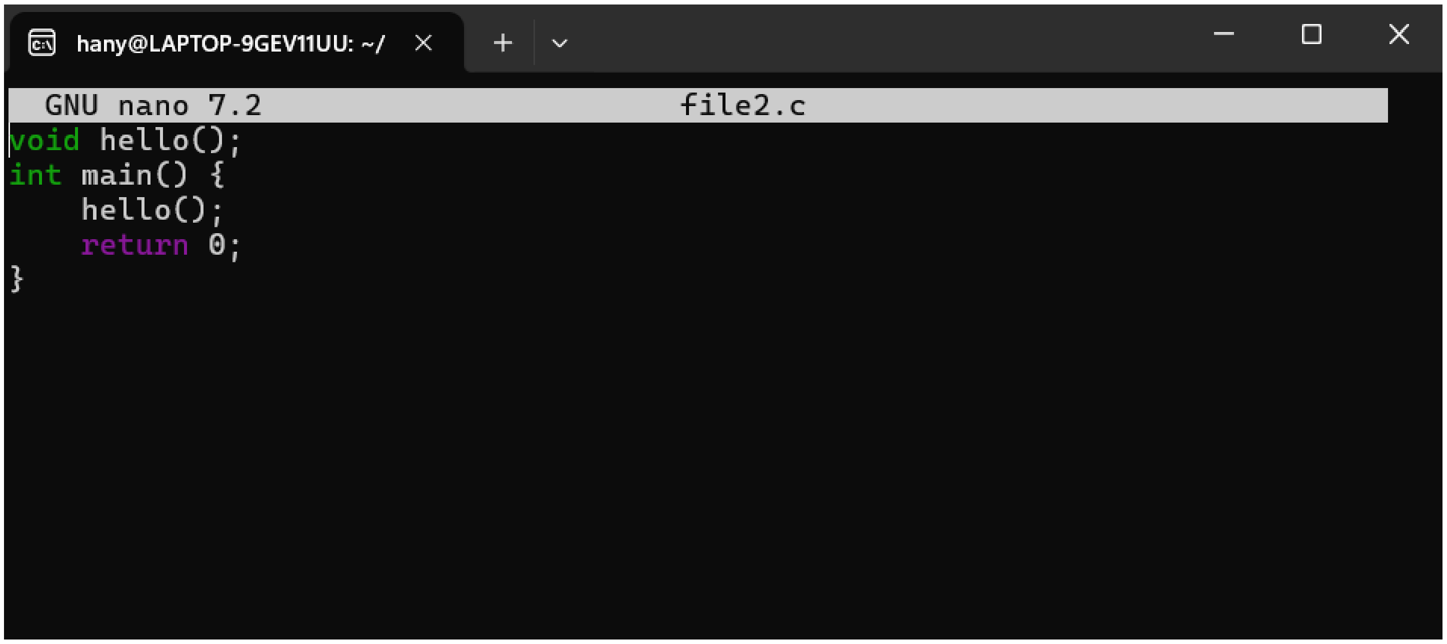
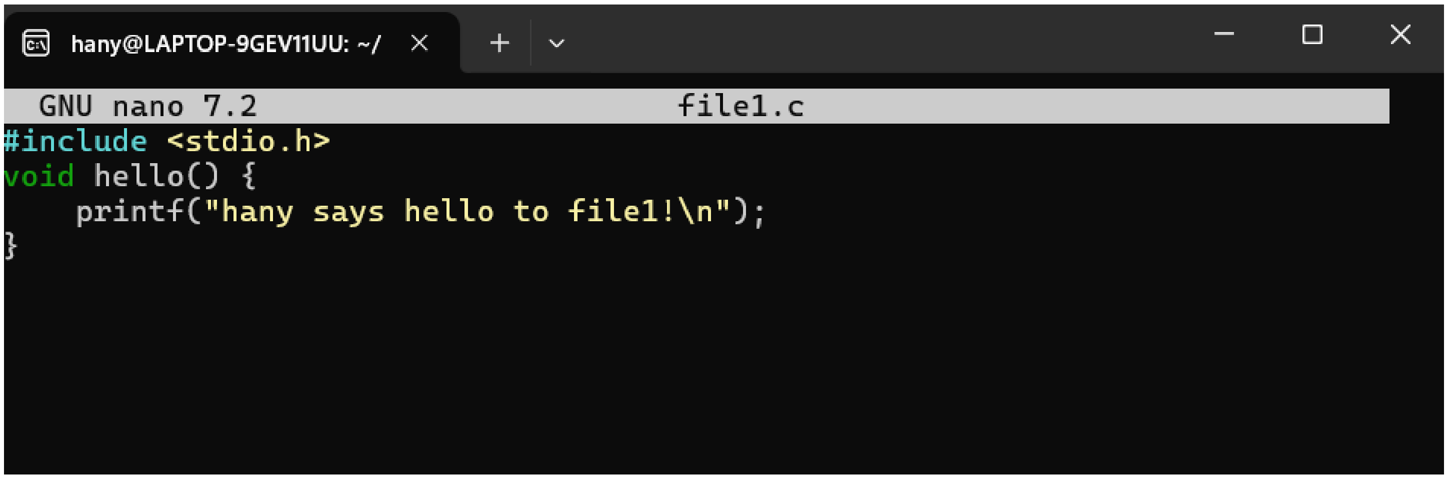
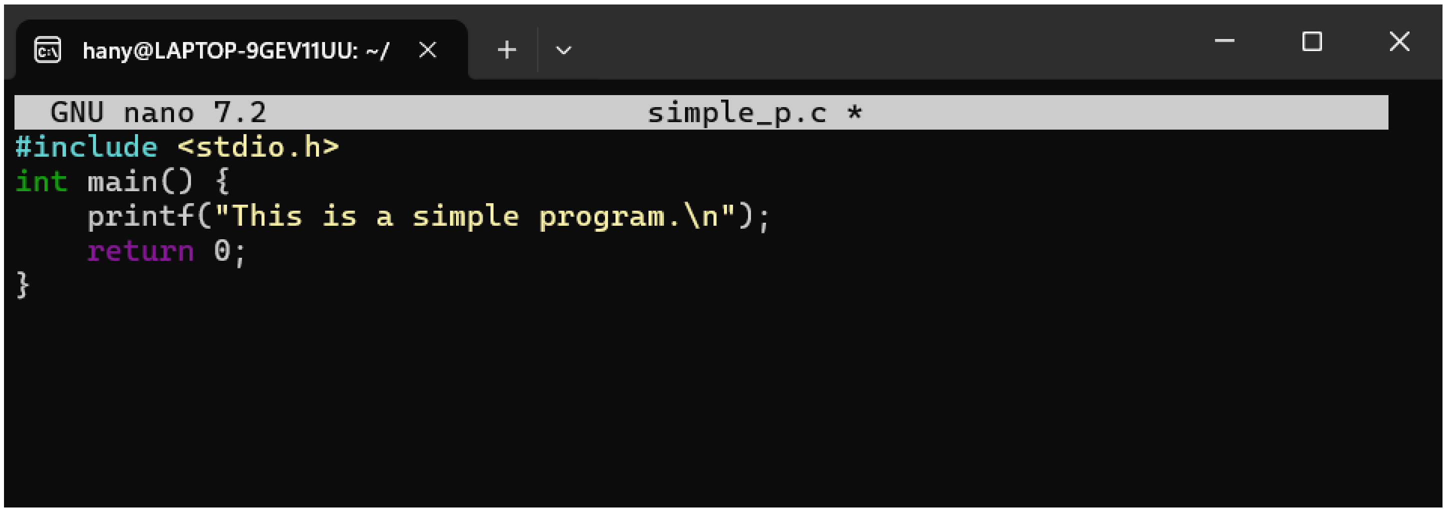


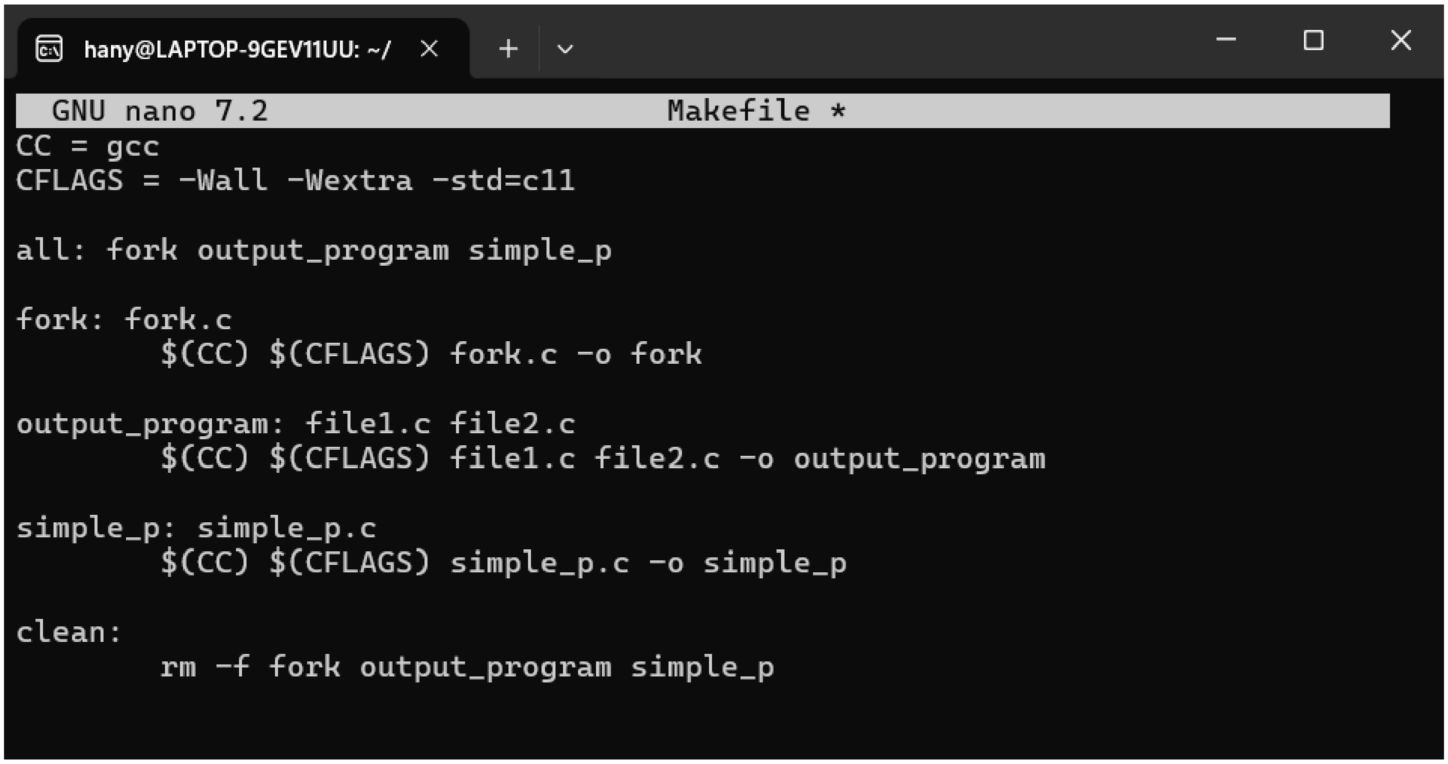
```
  GNU nano 7.2                          file2.c
void hello();
int main() {
    hello();
    return 0;
}
```



```
  GNU nano 7.2                          file1.c
#include <stdio.h>
void hello() {
    printf("hany says hello to file1!\n");
}
```

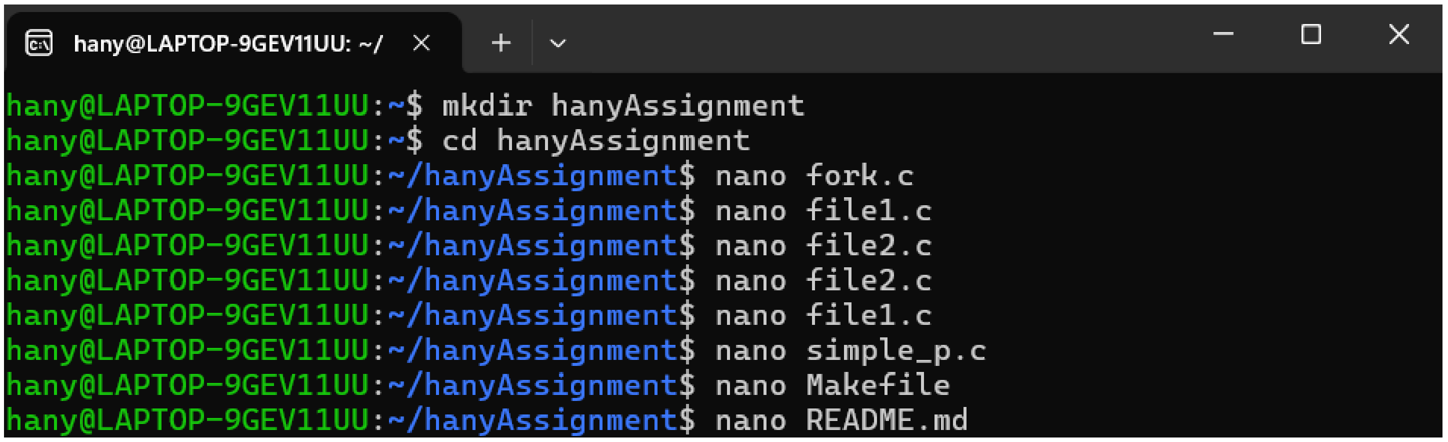


```
  GNU nano 7.2                          simple_p.c *
#include <stdio.h>
int main() {
    printf("This is a simple program.\n");
    return 0;
}
```



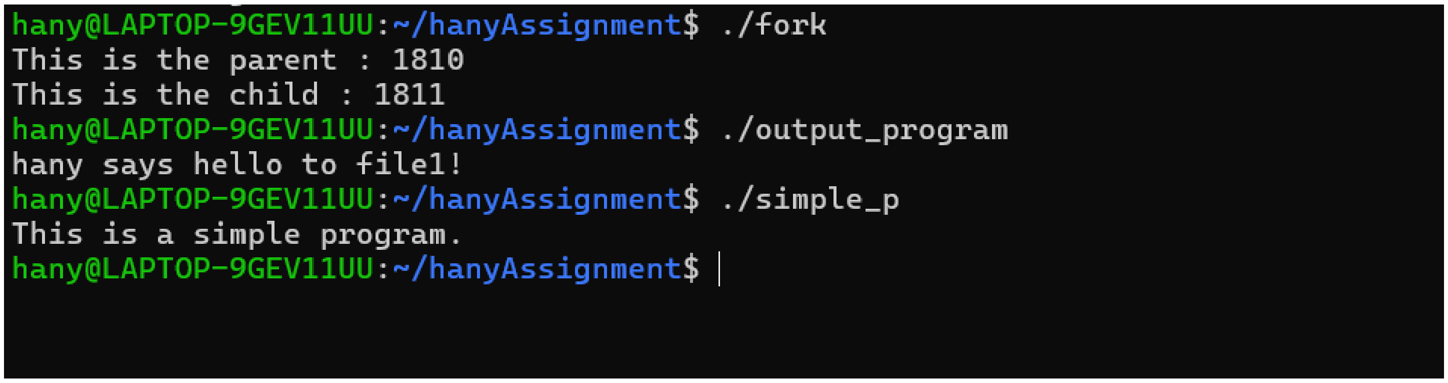
hany@LAPTOP-9GEV11UU: ~/

```
  GNU nano 7.2                          Makefile *
CC = gcc
CFLAGS = -Wall -Wextra -std=c11

all: fork output_program simple_p

fork: fork.c
        $(CC) $(CFLAGS) fork.c -o fork

output_program: file1.c file2.c
        $(CC) $(CFLAGS) file1.c file2.c -o output_program

simple_p: simple_p.c
        $(CC) $(CFLAGS) simple_p.c -o simple_p

clean:
        rm -f fork output_program simple_p
```

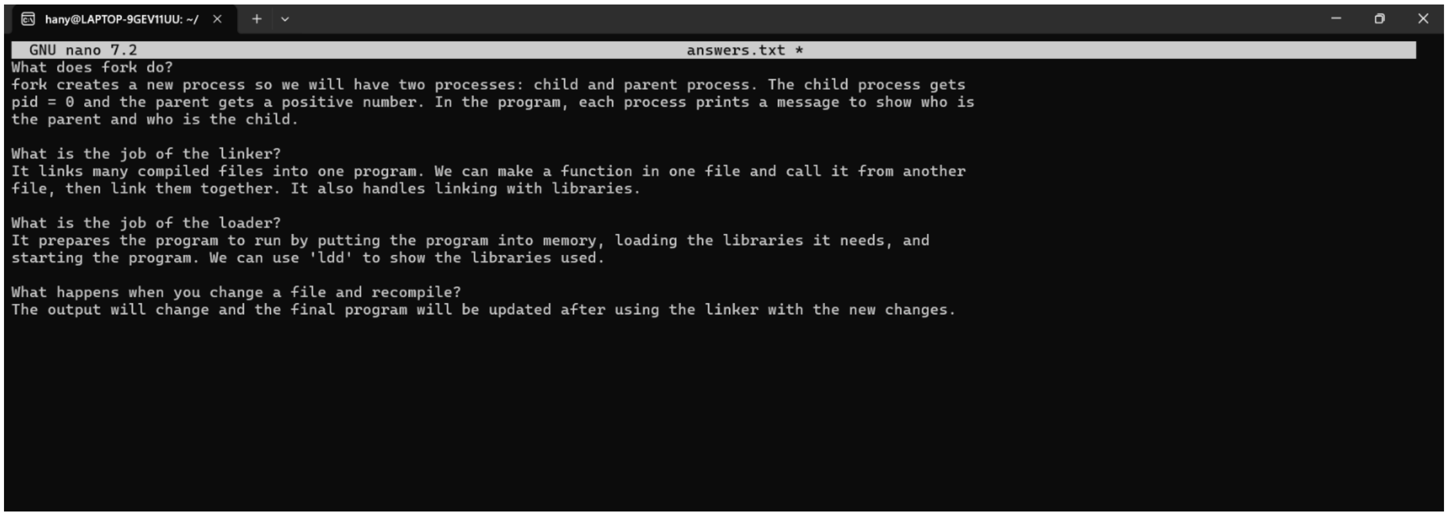


hany@LAPTOP-9GEV11UU: ~/

```
hany@LAPTOP-9GEV11UU:~$ mkdir hanyAssignment
hany@LAPTOP-9GEV11UU:~$ cd hanyAssignment
hany@LAPTOP-9GEV11UU:~/hanyAssignment$ nano fork.c
hany@LAPTOP-9GEV11UU:~/hanyAssignment$ nano file1.c
hany@LAPTOP-9GEV11UU:~/hanyAssignment$ nano file2.c
hany@LAPTOP-9GEV11UU:~/hanyAssignment$ nano file2.c
hany@LAPTOP-9GEV11UU:~/hanyAssignment$ nano file1.c
hany@LAPTOP-9GEV11UU:~/hanyAssignment$ nano simple_p.c
hany@LAPTOP-9GEV11UU:~/hanyAssignment$ nano Makefile
hany@LAPTOP-9GEV11UU:~/hanyAssignment$ nano README.md
```

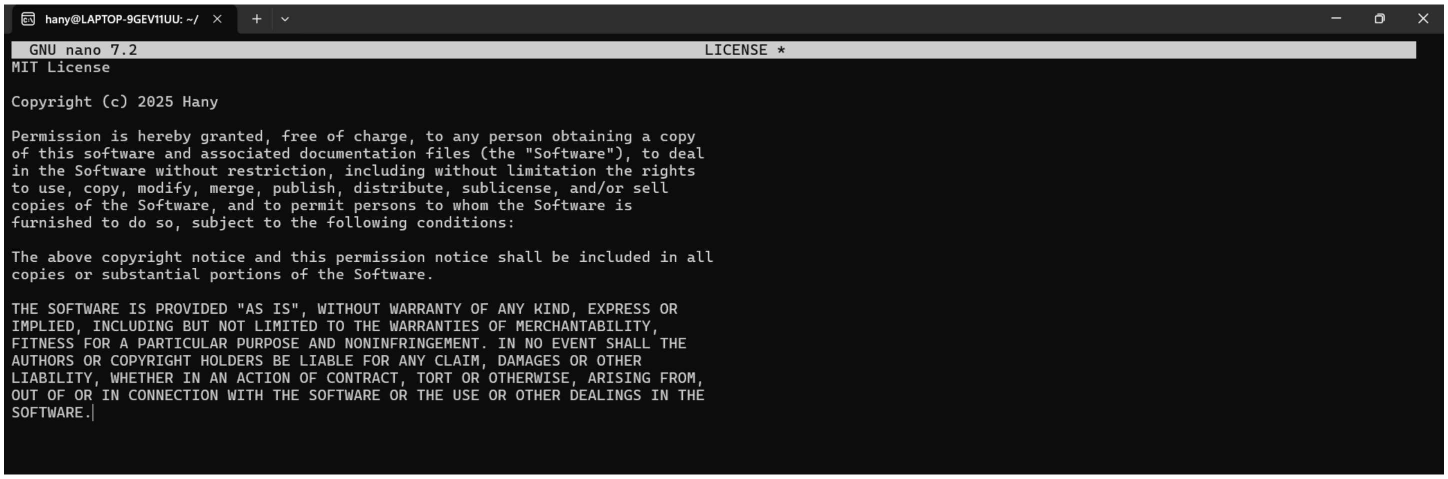


```
hany@LAPTOP-9GEV11UU:~/hanyAssignment$ ./fork
This is the parent : 1810
This is the child : 1811
hany@LAPTOP-9GEV11UU:~/hanyAssignment$ ./output_program
hany says hello to file1!
hany@LAPTOP-9GEV11UU:~/hanyAssignment$ ./simple_p
This is a simple program.
hany@LAPTOP-9GEV11UU:~/hanyAssignment$ 
```



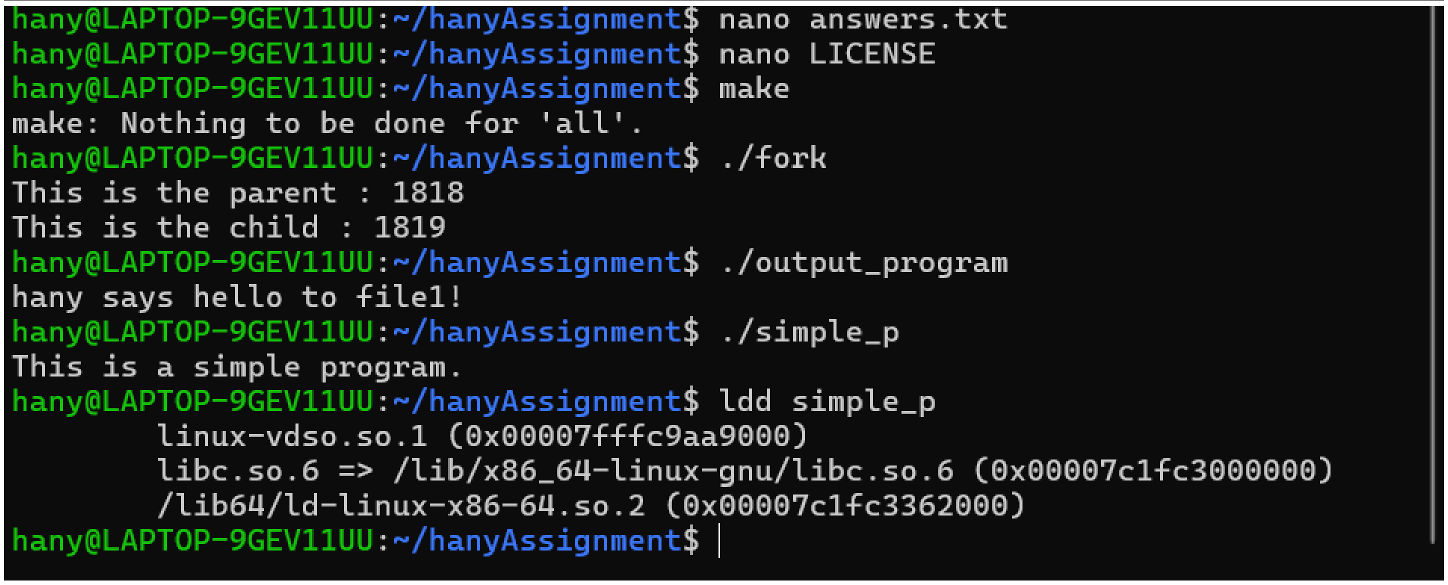
hany@LAPTOP-9GEV11UU: ~/

```
 GNU nano 7.2                                    answers.txt *
What does fork do?
fork creates a new process so we will have two processes: child and parent process. The child process gets
pid = 0 and the parent gets a positive number. In the program, each process prints a message to show who is
the parent and who is the child.

What is the job of the linker?
It links many compiled files into one program. We can make a function in one file and call it from another
file, then link them together. It also handles linking with libraries.

What is the job of the loader?
It prepares the program to run by putting the program into memory, loading the libraries it needs, and
starting the program. We can use 'ldd' to show the libraries used.

What happens when you change a file and recompile?
The output will change and the final program will be updated after using the linker with the new changes.
```

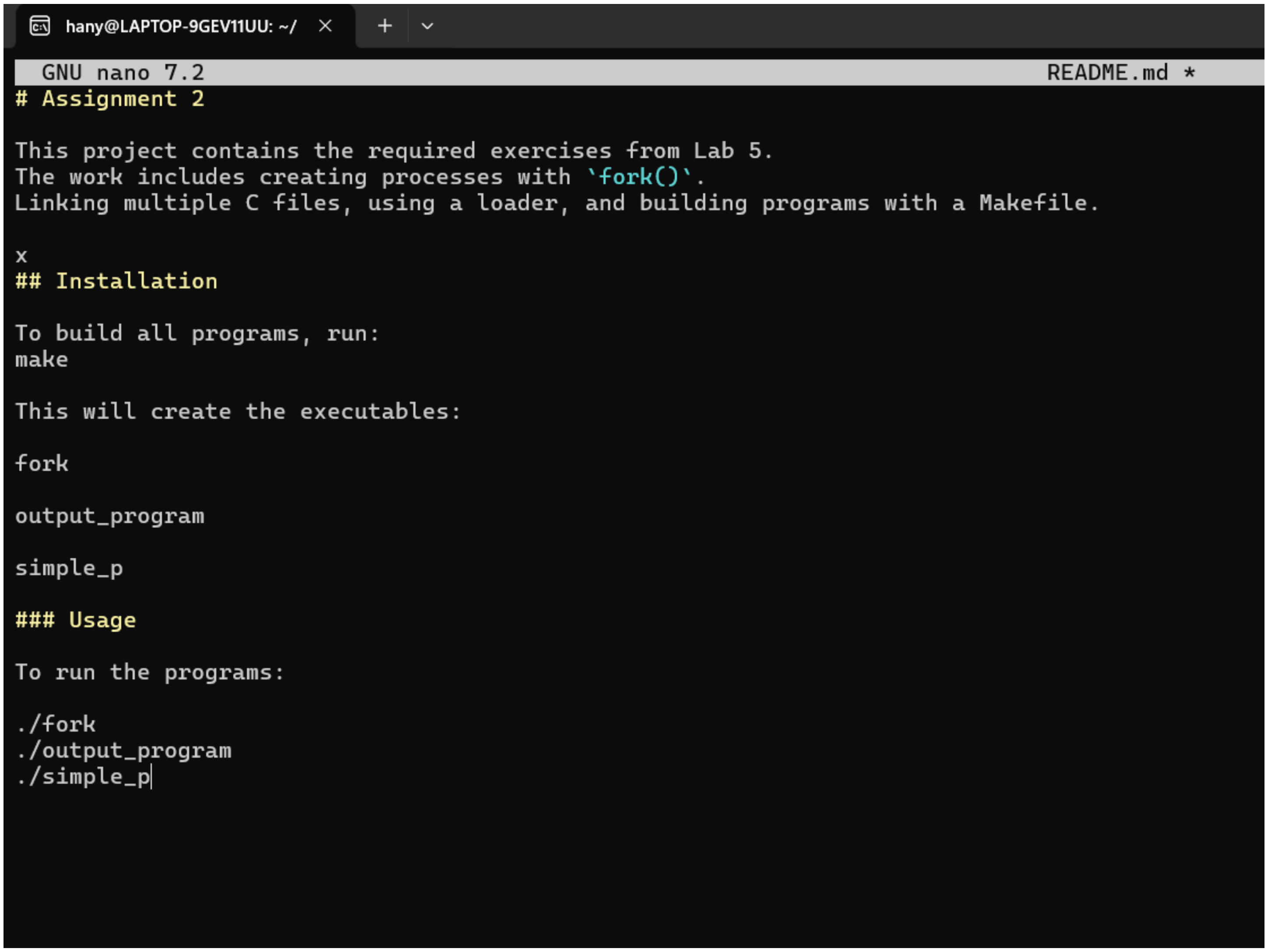


hany@LAPTOP-9GEV11UU: ~/

```
 GNU nano 7.2                                    LICENSE *
MIT License

Copyright (c) 2025 Hany

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

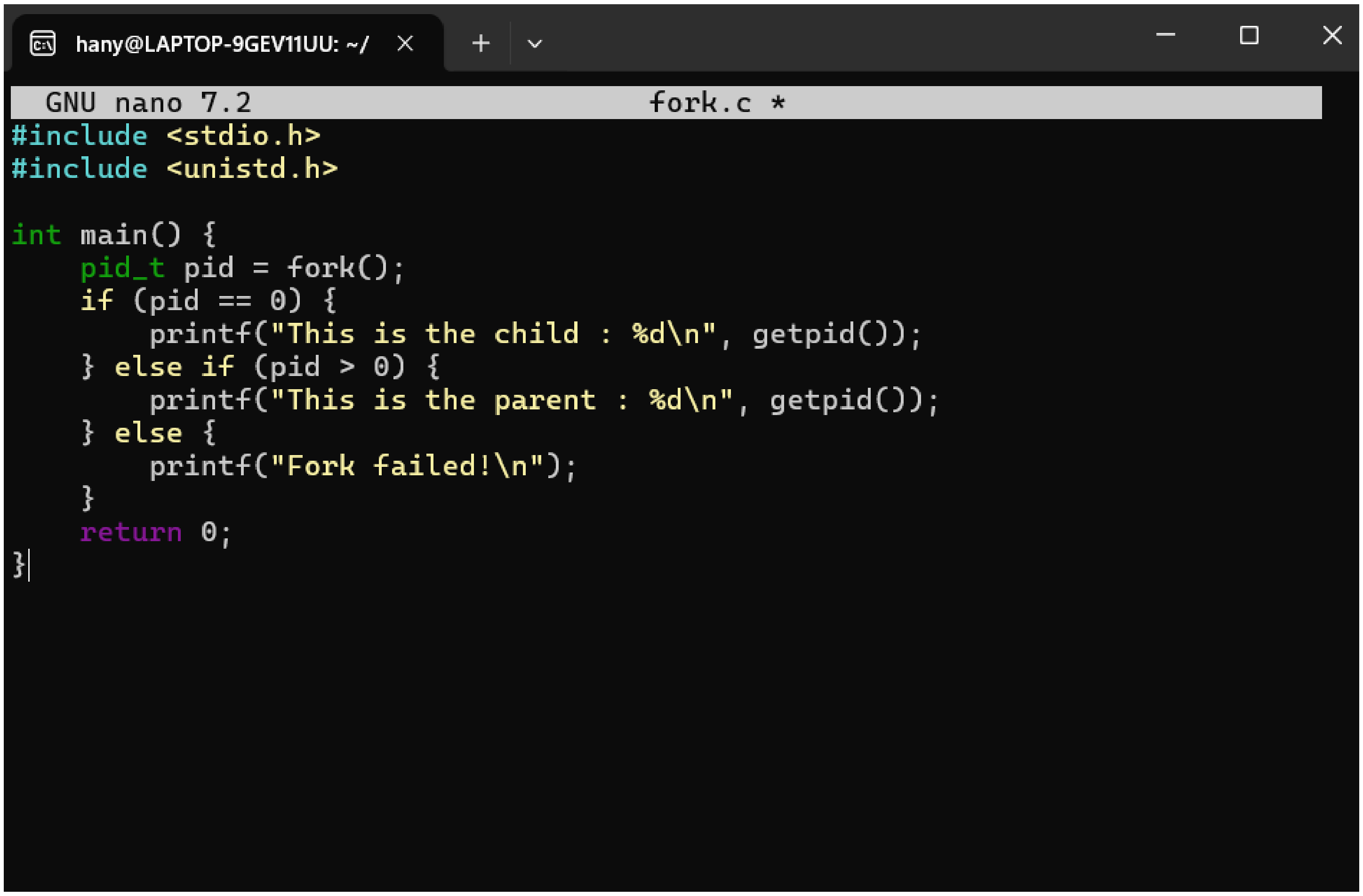


```
hany@LAPTOP-9GEV11UU:~/hanyAssignment$ nano answers.txt
hany@LAPTOP-9GEV11UU:~/hanyAssignment$ nano LICENSE
hany@LAPTOP-9GEV11UU:~/hanyAssignment$ make
make: Nothing to be done for 'all'.
hany@LAPTOP-9GEV11UU:~/hanyAssignment$ ./fork
This is the parent : 1818
This is the child : 1819
hany@LAPTOP-9GEV11UU:~/hanyAssignment$ ./output_program
hany says hello to file1!
hany@LAPTOP-9GEV11UU:~/hanyAssignment$ ./simple_p
This is a simple program.
hany@LAPTOP-9GEV11UU:~/hanyAssignment$ ldd simple_p
        linux-vdso.so.1 (0x00007fffc9aa9000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007c1fc3000000)
        /lib64/ld-linux-x86-64.so.2 (0x00007c1fc3362000)
hany@LAPTOP-9GEV11UU:~/hanyAssignment$
```



hany@LAPTOP-9GEV11UU: ~/

```
  GNU nano 7.2                                                           README.md *
# Assignment 2

This project contains the required exercises from Lab 5.
The work includes creating processes with `fork()`.
Linking multiple C files, using a loader, and building programs with a Makefile.

x
## Installation

To build all programs, run:
make

This will create the executables:

fork

output_program

simple_p

### Usage

To run the programs:

./fork
./output_program
./simple_p
```



```
  GNU nano 7.2                         fork.c *
#include <stdio.h>
#include <unistd.h>

int main() {
    pid_t pid = fork();
    if (pid == 0) {
        printf("This is the child : %d\n", getpid());
    } else if (pid > 0) {
        printf("This is the parent : %d\n", getpid());
    } else {
        printf("Fork failed!\n");
    }
    return 0;
}
```