

# **РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**

Факультет физико-математических и естественных наук Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ по лабораторной работе № 14

дисциплина:Операционные системы

- Студент: Гань Чжаолун
- Группа:НФИБД-01-21
- № ст. билета: 1032198038

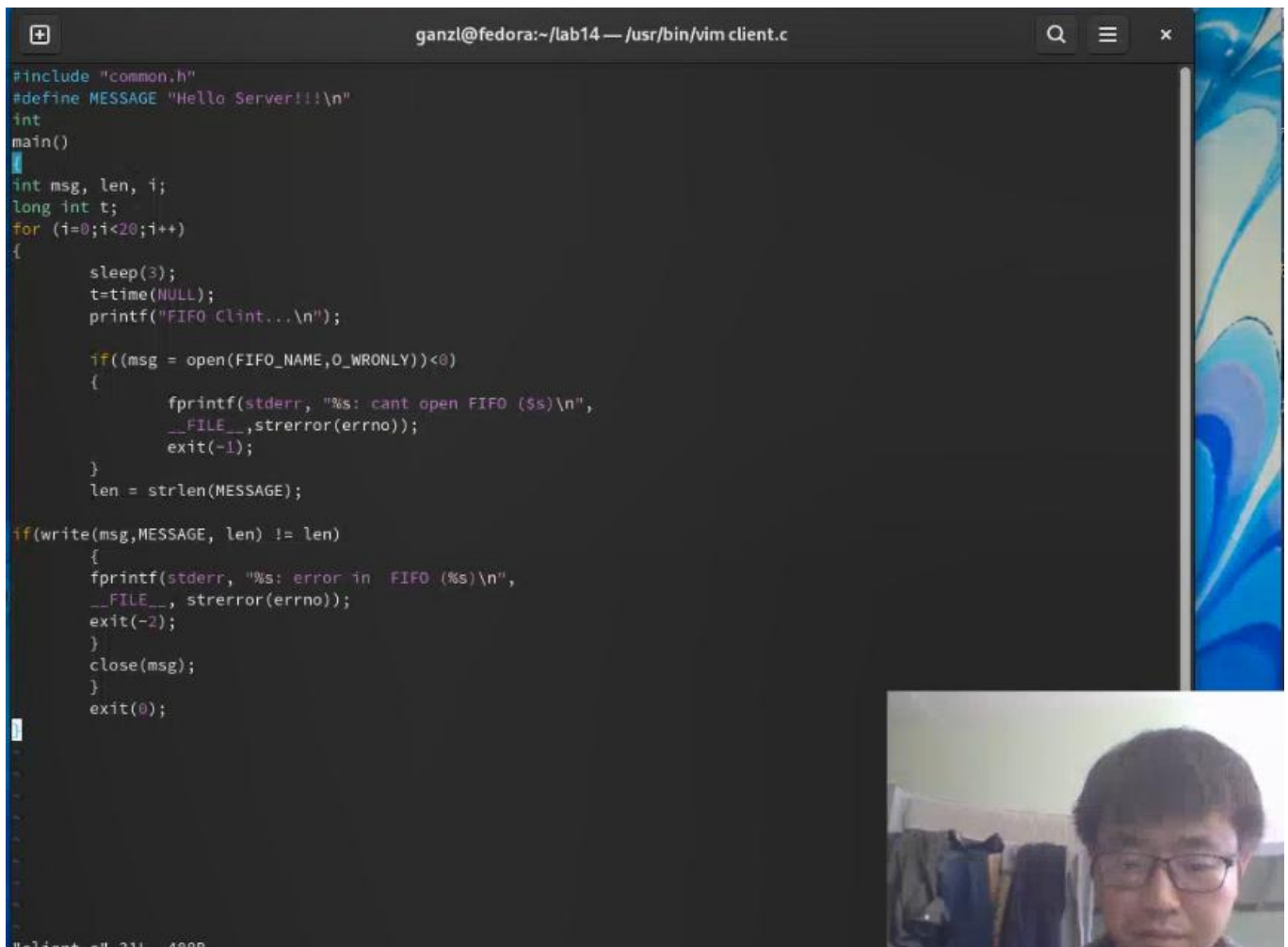
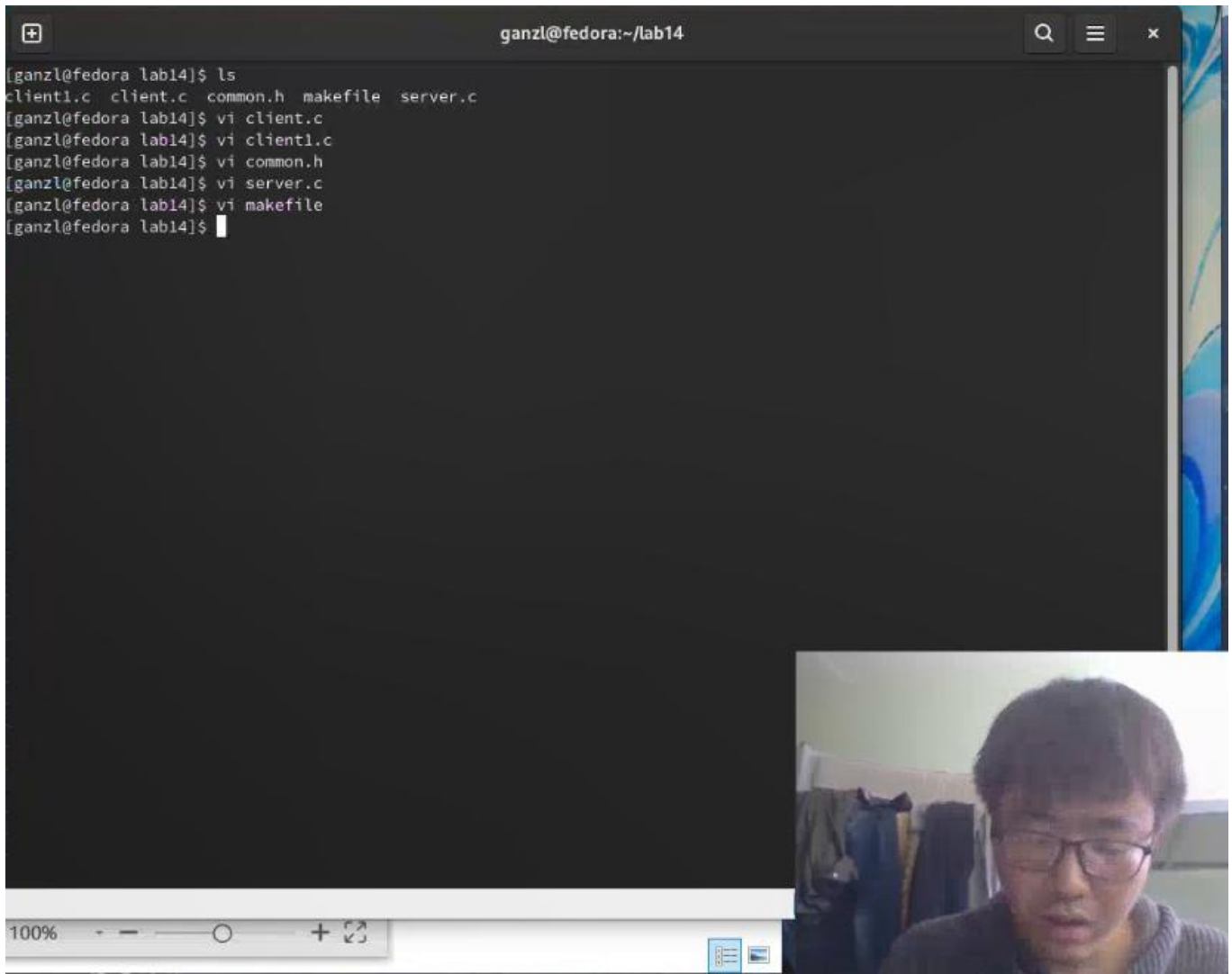
МОСКВА 2022 г.

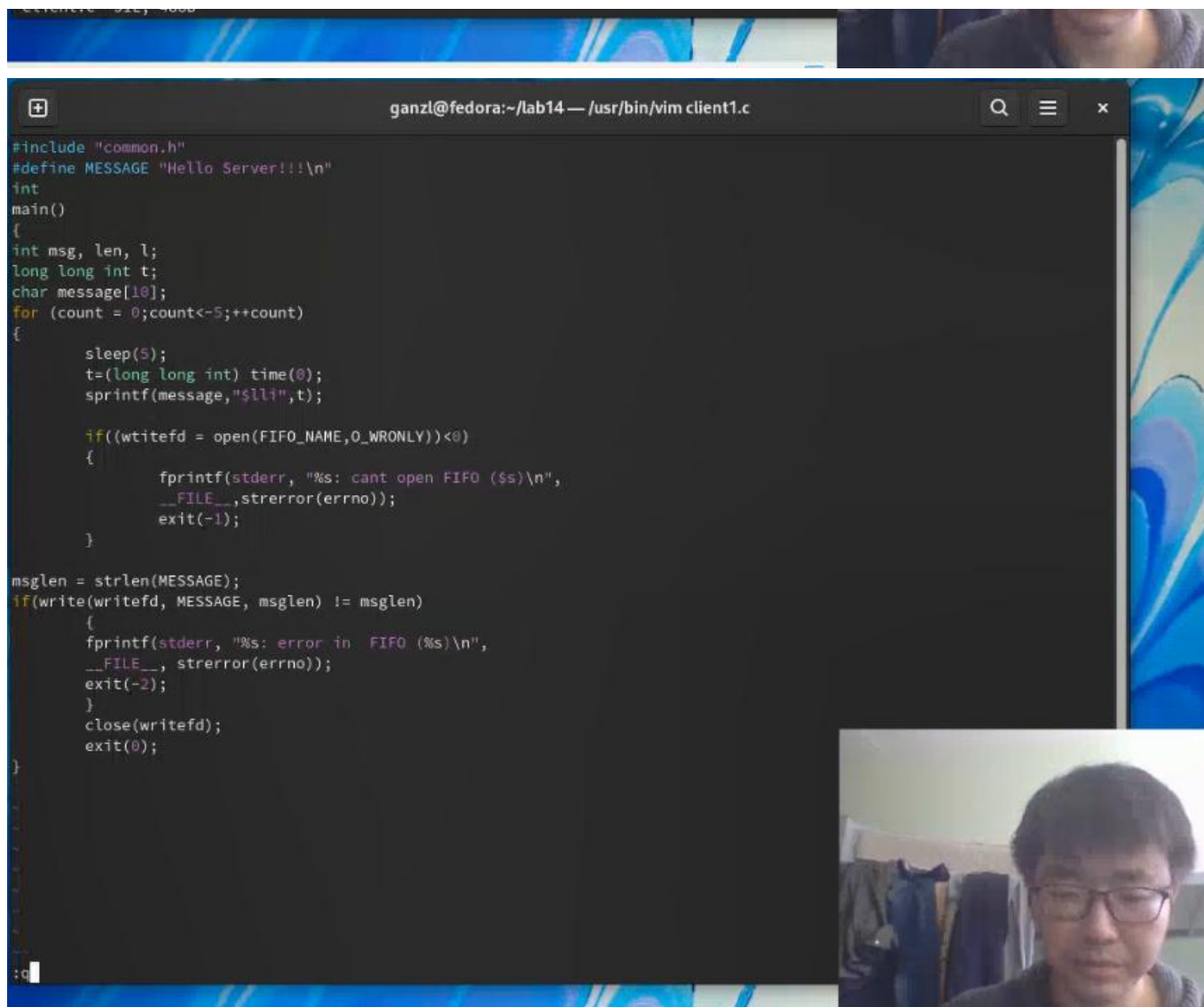
## **Цель работы:**

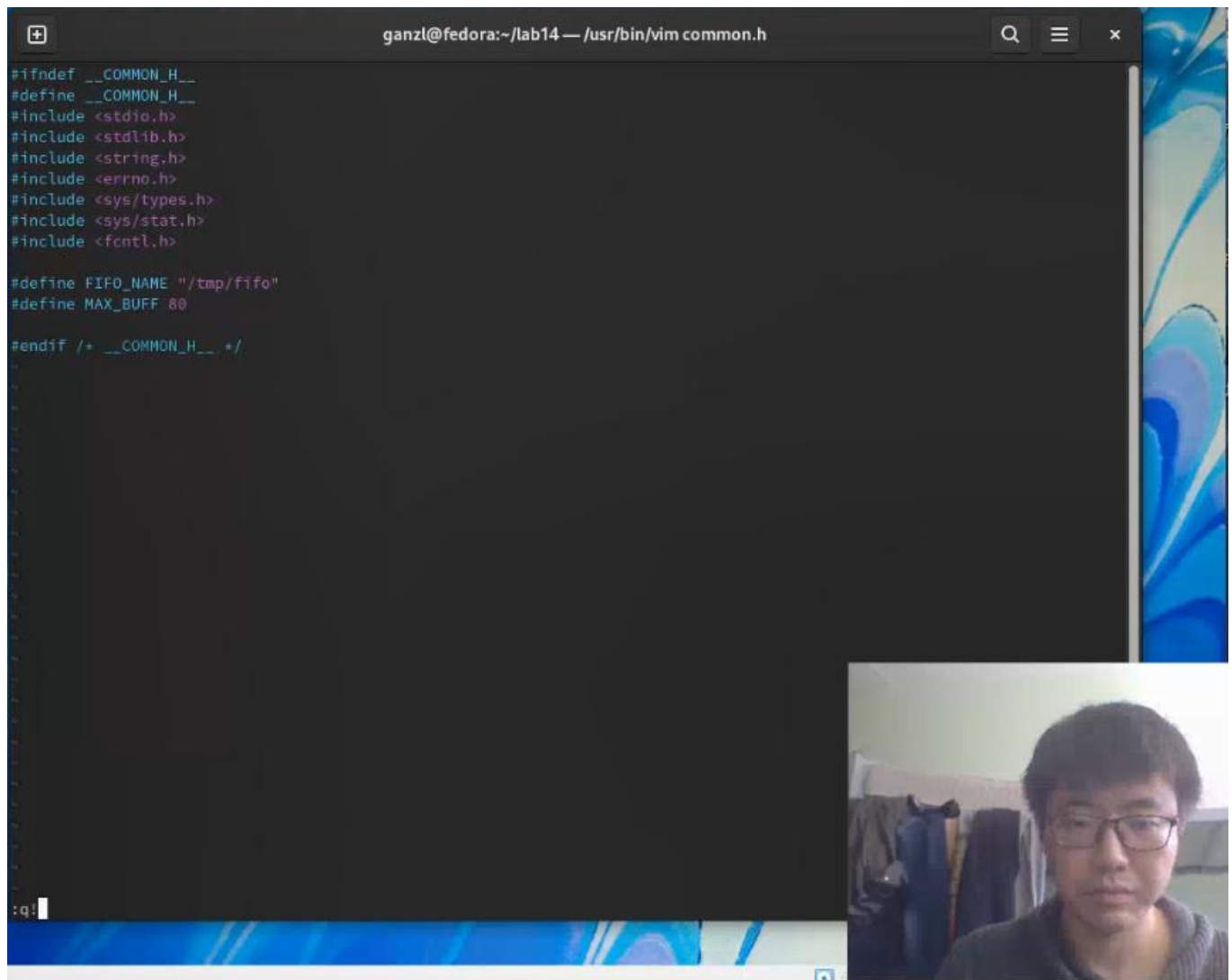
Приобретение практических навыков работы с именованными каналами.

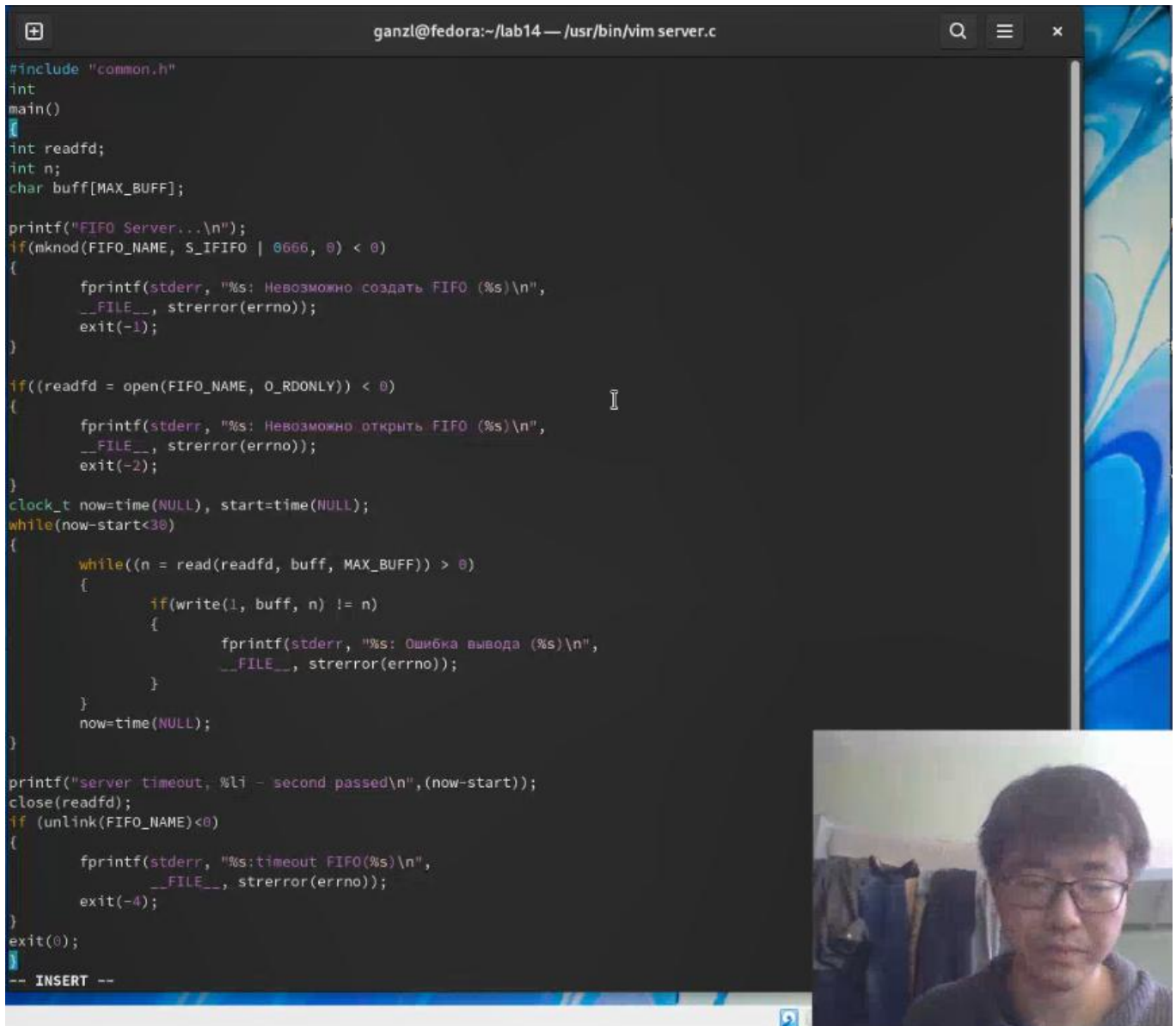
## **Выполнение работы:**

Сначала я создаю 5 файлов, показанных на рисунке, и редактирую их









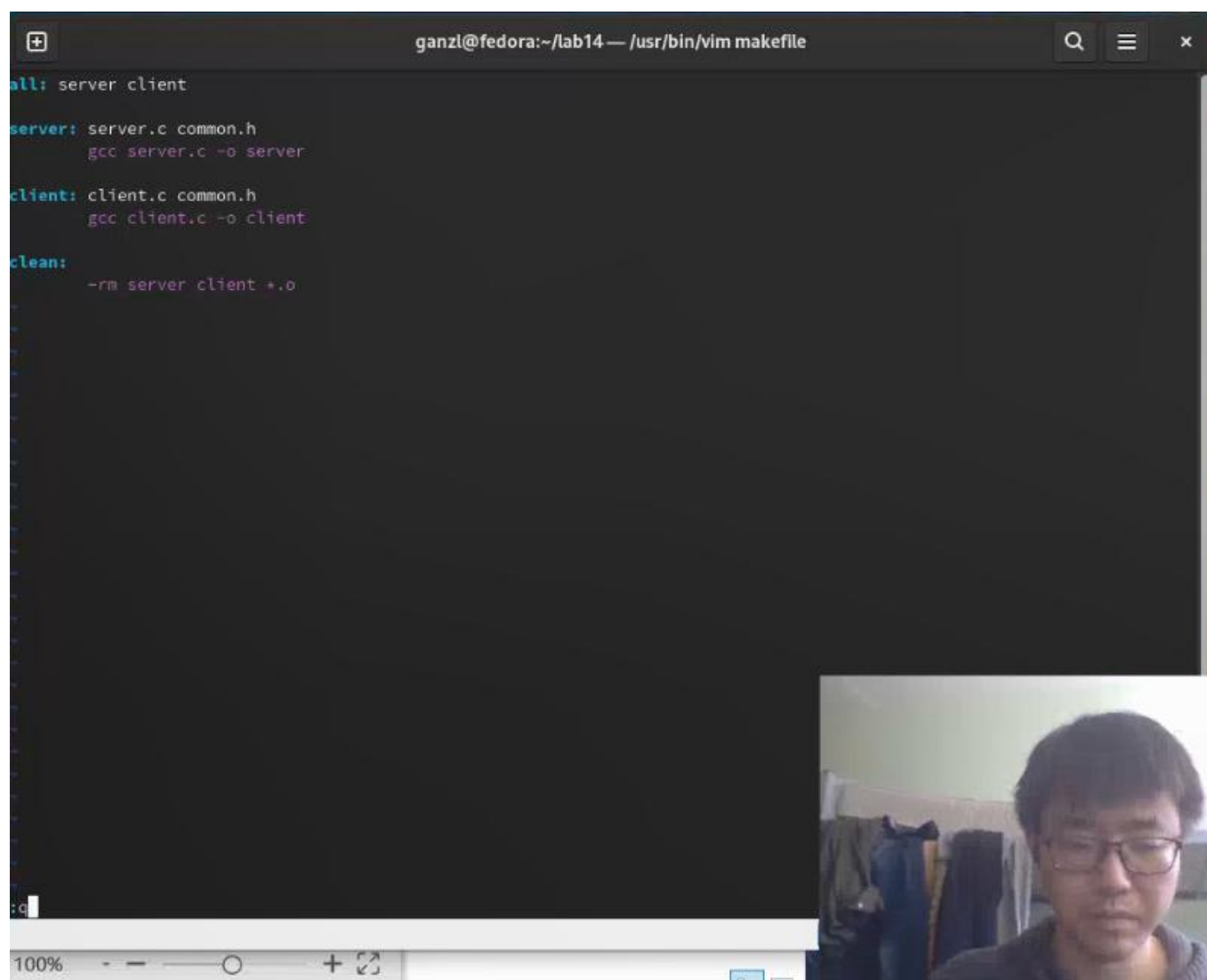
The screenshot shows a terminal window with a dark background and light-colored text. The terminal title bar reads "ganzl@fedora:~/lab14 — /usr/bin/vim server.c". The code is a C program for a FIFO server. It includes "common.h", defines "int", and has a "main()" function. It declares "int readfd;", "int n;", and "char buff[MAX\_BUFF];". It prints "FIFO Server...\n". It checks if "mknod(FIFO\_NAME, S\_IFIFO | 0666, 0) < 0". If true, it prints an error "Невозможно создать FIFO (%s)\n" and exits with -1. It then checks if "open(FIFO\_NAME, O\_RDONLY) < 0". If true, it prints an error "Невозможно открыть FIFO (%s)\n" and exits with -2. It sets a timeout of 30 seconds using "clock\_t" and "time". It enters a "while" loop that runs as long as "now-start < 30". Inside, it reads from "readfd" into "buff". If "write(1, buff, n) != n", it prints an error "Ошибка вывода (%s)\n" and continues. It then updates "now=time(NULL)". After the loop, it prints "server timeout, %li - second passed\n", closes "readfd", and checks if "unlink(FIFO\_NAME) < 0". If true, it prints an error "timeout FIFO(%s)\n" and exits with -4. Finally, it exits with 0. The terminal shows the "-- INSERT --" prompt at the bottom. In the bottom right corner, there is a small video inset showing a person with glasses and a beard, looking at the camera.

```
#include "common.h"
int
main()
{
    int readfd;
    int n;
    char buff[MAX_BUFF];

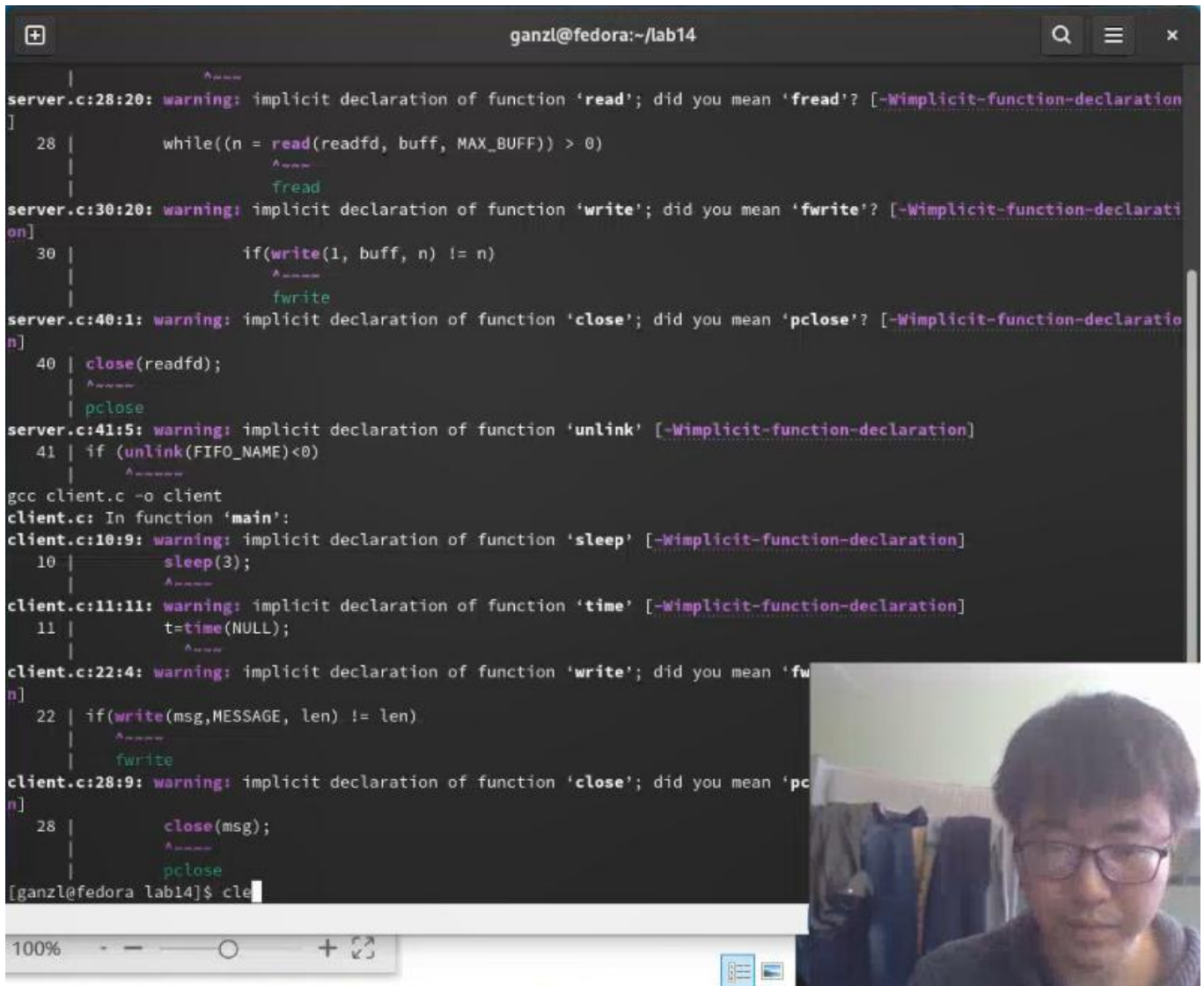
    printf("FIFO Server...\n");
    if(mknod(FIFO_NAME, S_IFIFO | 0666, 0) < 0)
    {
        fprintf(stderr, "%s: Невозможно создать FIFO (%s)\n",
            __FILE__, strerror(errno));
        exit(-1);
    }

    if((readfd = open(FIFO_NAME, O_RDONLY)) < 0)
    {
        fprintf(stderr, "%s: Невозможно открыть FIFO (%s)\n",
            __FILE__, strerror(errno));
        exit(-2);
    }
    clock_t now=time(NULL), start=time(NULL);
    while(now-start<30)
    {
        while((n = read(readfd, buff, MAX_BUFF)) > 0)
        {
            if(write(1, buff, n) != n)
            {
                fprintf(stderr, "%s: Ошибка вывода (%s)\n",
                    __FILE__, strerror(errno));
            }
        }
        now=time(NULL);
    }

    printf("server timeout, %li - second passed\n", (now-start));
    close(readfd);
    if (unlink(FIFO_NAME)<0)
    {
        fprintf(stderr, "%s: timeout FIFO(%s)\n",
            __FILE__, strerror(errno));
        exit(-4);
    }
    exit(0);
}
-- INSERT --
```



Затем я использовал команду `make` для создания файла выполнения



```
ganzl@fedora:~/lab14
server.c:28:20: warning: implicit declaration of function 'read'; did you mean 'fread'? [-Wimplicit-function-declaration]
28 |         while((n = read(readfd, buff, MAX_BUFF)) > 0)
   |                   ^~~~~
   |                   fread
server.c:30:20: warning: implicit declaration of function 'write'; did you mean 'fwrite'? [-Wimplicit-function-declaration]
30 |         if(write(1, buff, n) != n)
   |            ^~~~~
   |            fwrite
server.c:40:1: warning: implicit declaration of function 'close'; did you mean 'pclose'? [-Wimplicit-function-declaration]
40 | close(readfd);
   | ^~~~~
   | pclose
server.c:41:5: warning: implicit declaration of function 'unlink' [-Wimplicit-function-declaration]
41 | if (unlink(FIFO_NAME)<0)
   | ^~~~~
gcc client.c -o client
client.c: In function 'main':
client.c:10:9: warning: implicit declaration of function 'sleep' [-Wimplicit-function-declaration]
10 |     sleep(3);
   |     ^~~~~
client.c:11:11: warning: implicit declaration of function 'time' [-Wimplicit-function-declaration]
11 |     t=time(NULL);
   |     ^~~~~
client.c:22:4: warning: implicit declaration of function 'write'; did you mean 'fwrite'? [-Wimplicit-function-declaration]
22 | if(write(msg,MESSAGE, len) != len)
   |    ^~~~~
   |    fwrite
client.c:28:9: warning: implicit declaration of function 'close'; did you mean 'pclose'? [-Wimplicit-function-declaration]
28 |     close(msg);
   |     ^~~~~
   |     pclose
[ganzl@fedora lab14]$ cle
```

100% — — — — — + ↻

Наконец, я запустил сервер и клиент, показав, что ссылка прошла успешно, и сервер автоматически закрылся через 30 секунд.



```

ganzl@fedora:~/lab14
[ganzl@fedora lab14]$ ls
client client.c client.c common.h makefile server server
[ganzl@fedora lab14]$ ./server
FIFO Server...
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
server timeout, 30 - second passed
[ganzl@fedora lab14]$

ganzl@fedora:~/lab14
[ganzl@fedora lab14]$ ./client
FIFO Clint...
FIFO Clint...
FIFO Clint...
FIFO Clint...
FIFO Clint...
FIFO Clint...
FIFO Clint...
FIFO Clint...
FIFO Clint...
FIFO Clint...
FIFO Clint...
FIFO Clint...
FIFO Clint...
FIFO Clint...
FIFO Clint...
client.c: cant open FIFO ($s)
[ganzl@fedora lab14]$

```

## Контрольные вопросы:

1. В чем ключевое отличие именованных каналов от неименованных?
  - Именованные каналы, в отличие от неименованных, могут использоваться неродственными процессами. Они дают вам, по сути, те же возможности, что и неименованные каналы, но с некоторыми преимуществами, присущими обычным файлам. Именованные каналы используют специальную запись в директории для управления правами доступа.
2. Возможно ли создание неименованного канала из командной строки?
  - можно
3. Возможно ли создание именованного канала из командной строки?
  - можно
4. Опишите функцию языка C, создающую неименованный канал.
  - mknod
5. Опишите функцию языка C, создающую именованный канал.
  - mkfifo
6. Что будет в случае прочтения из fifo меньшего числа байтов, чем находится в канале? Большого числа байтов?
  - 1. При чтении меньшего числа байтов, чем находится в канале или FIFO, возвращается требуемое число байтов, остаток сохраняется для последующих чтений.
  - 2. При чтении большего числа байтов, чем находится в канале или FIFO, возвращается доступное число байтов. Процесс, читающий из канала, должен соответствующим образом обработать ситуацию, когда прочитано меньше, чем заказано.



7. Аналогично, что будет в случае записи в fifo меньшего числа байтов, чем позволяет буфер? Большого числа байтов?

- 1. Запись числа байтов, меньшего емкости канала или FIFO, гарантированно атомарно. Это означает, что в случае, когда несколько процессов одновременно записывают в канал, порции данных от этих процессов не перемешиваются.
- 2. При записи большего числа байтов, чем это позволяет канал или FIFO, вызов `write(2)` блокируется до освобождения требуемого места. При этом атомарность операции не гарантируется. Если процесс пытается записать данные в канал, не открытый ни одним процессом на чтение, процессу генерируется сигнал `SIGPIPE`, а вызов `write(2)` возвращает 0 с установкой ошибки (`errno=ERRPIPE`) (если процесс не установил обработки сигнала `SIGPIPE`, производится обработка по умолчанию — процесс завершается).

8. Могут ли два и более процессов читать или записывать в канал?

- можно

9. Опишите функцию `write` (тип возвращаемого значения, аргументы и логику работы). Что означает 1 (единица) в вызове этой функции в программе `server.c` (строка 42)?

- Отправить сообщение на сервер 10. Опишите функцию `strerror`.
- Строковая функция `strerror` - функция языков C/C++, транслирующая код ошибки, который обычно хранится в глобальной переменной `errno`, в сообщение об ошибке, понятном человеку.

## Вывод:

- Я научился практическим навыкам работы с назначенными каналами.