Assignment 2
CN2020

Assignment 2 Announcement

Specification (1/4)

- *In this assignment, you need to implement a simple Network Storage System, with these functions:*
  - *Client can* **watch a ".mpg" video (streaming)** *on the server*
  - *Client can* **upload files** *to server*
  - *Client can* **download files** *from server*
  - *Client can* **list what files are in the folder of server**
- *For video steaming, you* **don't need to send audio**. *You can just send* **raw frames (or encoded frames)**.
- *Video streaming requires* **buffering** *at both client side and server side.* **(bonus)**
- *After upload and download, you need to ensure the* **files are identical between source and destination**.
- *In this assignment, all the transmission should be implemented by* **the socket of TCP**.

Specification (2/4)

- *You are required to write a Makefile for compilation. Thus, the command should be*

```
$ make client          // for client code
$ make server          // for server code
```

- *After compilation, there should be 2 binary files named "client" and "server"*
- *When we launch the server app, we will enter*

```
$ ./server   [port]          // [port] will be determined
```

- *When we launch the client app, we will enter*

```
$ ./client    [ip:port]      // ip is the ip address of server
                                port is determined by the
                                command above
```

- *After launching, client / server is required to create their own folders while any folder doesn't exist.*

Specification (3/4)

- *Server is required to support multiple connections. That is, there* **can be more than 1 client** *connecting to the server* **simultaneously**.
- *After building up of a connection, a user can enter the commands below on the client,*

```
$ ls                    // Then, the client's will print out what
                           files is in the server's folder.

$ put <filename>   // Then, the client will upload the file
                   with the <filename> to the server's
                           folder

$ get <filename>   // Then, the client will download the
                           file with the <filename> to the
                           client's folder

$ play <videofile> // Then, the client will play the
                           <videofile> from the buffer at server
                           side to the client
```

Specification (4/4)

- *If the command doesn't exist or the command format is wrong, please print out "***Command not found.***" or "***Command format error.***" on the client.*
- *If the file doesn't exist while putting or getting a file, please print out "***The '<filename>' doesn't exist.***" on the client.*
- *If the video file is not a ".mpg" file while playing a video file, please print out "***The '<videofile>' is not a mpg file.***"*
- ***Client should be able to send another command after a command is finished.***
- *The multiple connections should be implemented with **pthread.h**, while the video player should be implemented with **OpenCV**.*
- *The implementation must be in **C or C++**.*

Grading Policy (1/3)

- **This assignment accounts for 12% of the total score.**
- **Command Sending** **(10%)**
  - *The client sends commands correctly* *(5%)*
  - *The client prints out responses correctly* *(5%)*
- **Video Streaming** **(25%)**
  - *Correctly play a **720p** Video (1280*720)* *(5%)*
  - *Correctly play a **resolution-unknown** video* *(10%)*

    *(That is, **client has no idea about the resolution** of the video before requesting a video to play.)*
  - *Correctly implement **frame encoding/decoding*** *(10%)*
    - *If you only send **raw frames*** *(5%)*
  - *(Bonus 10%) Correctly implement **buffering***
- **File Transferring** **(25%)**
  - *There would be 5 files with different sizes* *(5% * 5)*

  *(You will get **0 point** in a testcase if the transfer of a testcase is **terminated** or **halts** before finished, or the files are **not identical between source and destination** after the transfer.)*

Grading Policy (2/3)

- **Multiple Connections** **(20%)**
  - *Use <pthread.h> to achieve this function (basic)* *(10%)*
  - *Use select() to achieve this function (advance)* *(20%)*
  - *You just need to choose one of above to implement.*
- **Report** **(20%)**
  - *Draw a flowchart of the video streaming and explains how it works in detail.* *(5%)*
  - *Draw a flowchart of the file transferring and explains how it works in detail.* *(5%)*
  - *What is SIGPIPE? It is possible to happen to your code? If so, how do you handle it?* *(5%)*
  - *Is blocking I/O equal to synchronized I/O? Please give me some examples to explain it.* *(5%)*

Grading Policy (3/3)

- **Submission**
  - *Your report format must be in* **".pdf"** *format and named "report.pdf", or else* **you will get zero point** *in the part.*
  - *Please put all the files* **into a folder** *named* **hw2_<student id>**, *and compress the folder as a* **.zip** *file, and then submit the* **.zip** *file to NTU Cool. The filename is* **hw2_<student id>.zip** *(alphabets in lowercase).*
  - *If we* **cannot compile or execute your code**, *you will* **have a chance to demo your results** *in your own environment.*
  - *The penalty for* **wrong format** *is* **10 points**.
  - **No plagiarism is allowed. A plagiarist will be graded zero.**
- **Deadline**
  - *Deadline:* **23:59:59, December 15th, 2020 (get * 0.9 points)**
  - *Penalty for late submission is* **"10% per day"** .

Environment Setup

Environment

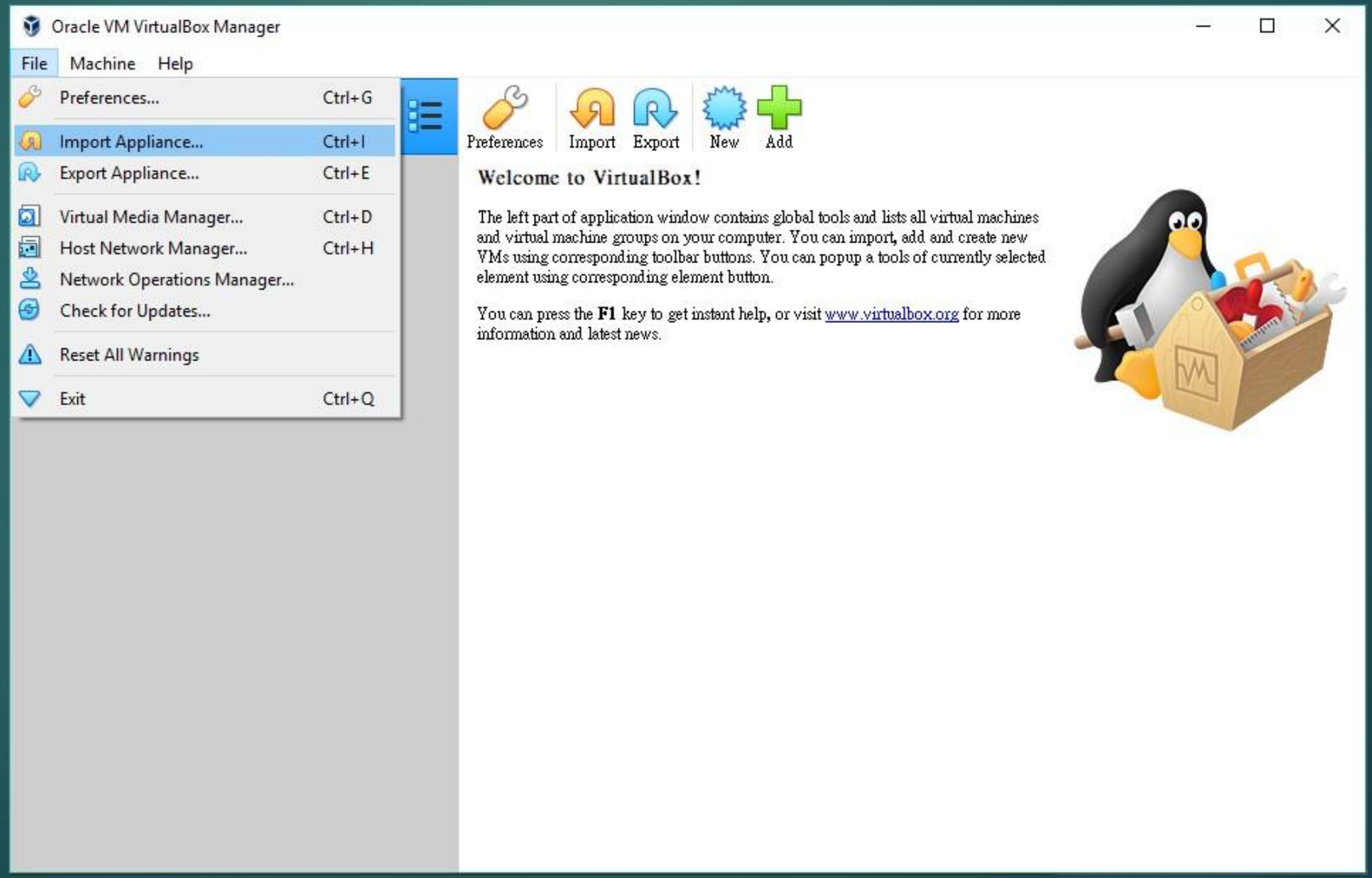
- *We provide a VirtualBox environment for you to run our binary code and you can run Wireshark on this environment.*
- *If you would like to setup the environment on your OS rather than our virtual machine, here is information of our environment*
  - **Ubuntu 16.04 x64**
  - **OpenCV 3.3.1 (will be also required in later Assignments)**
- *You can install OpenCV 3.3.1 by following the instruction here.*

VirtualBox Setup

- *Download the **VM** from*
  - *our server*
  - *our Google Drive*
- *Install Virtualbox (natively installed on the computers of Lab R204).*

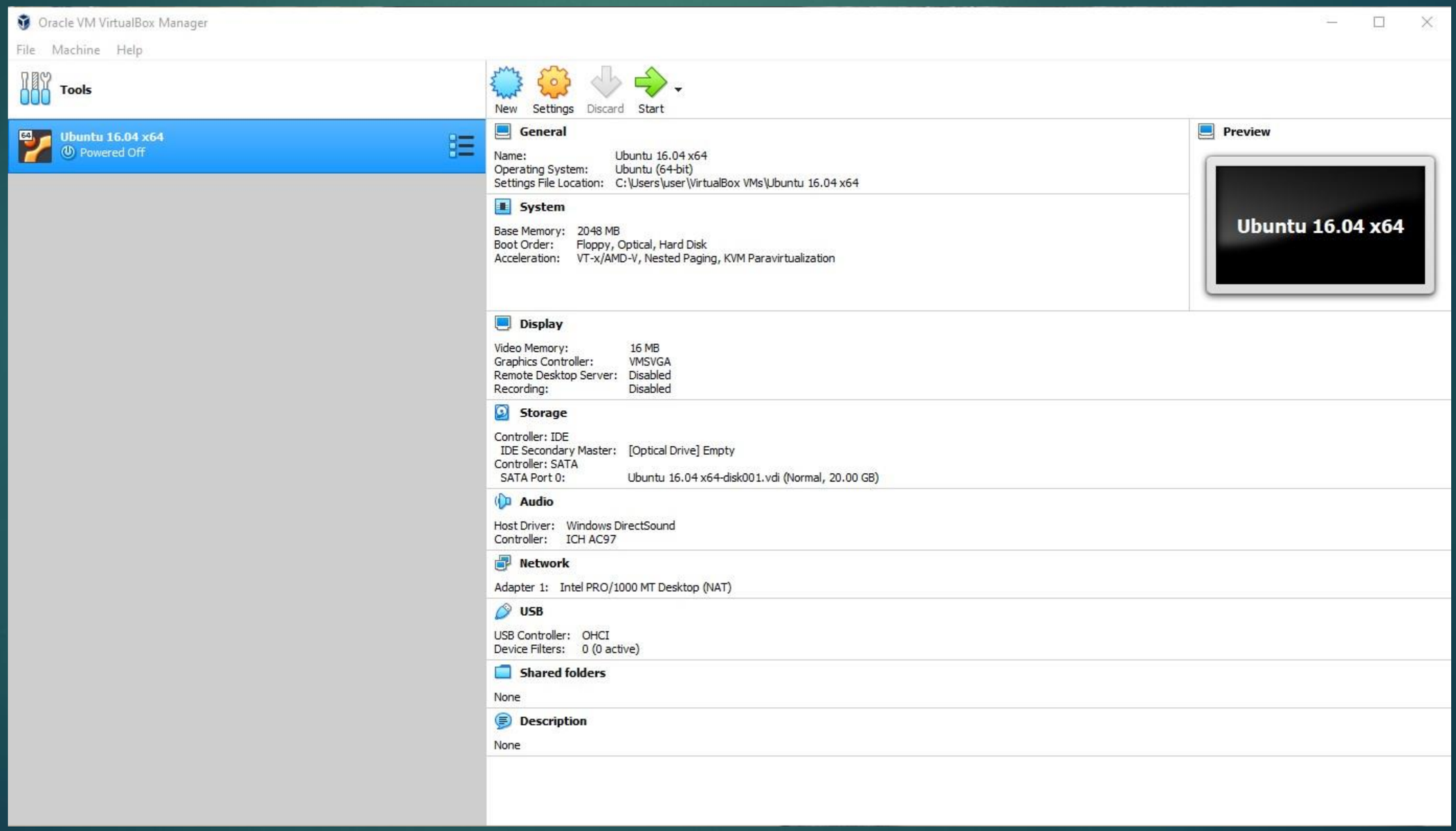
VirtualBox Setup

- *Go to "**File**" and click "**Import Appliance**" to import the "**CN-Ubuntu_16.04_x64.ova**"*



VirtualBox Setup

- *Choose "**Ubuntu 16.04 x64**" and then start the machine.*



Auxiliary Libraries

Example code

SocketExample/

├── client.cpp
├── server.cpp
├── Makefile

// You have learned

├── tmp.mpg
├── pthread.cpp
├── openCV.cpp
└── ffmpegExample/
├── ffmpegExample.c
├── ffmpeg_install.sh

Pthread

- *Pthead, i.e., POSIX Thread, is used to implement multi-thread parallelization in POSIX environment.*
- *You can use pthread to achieve multiple connections.*
- *You don't need to deal with synchronization issues, i.e., in our testcases, it won't put a video with the same file name.*
- *An example code in pthread.cpp*
- *To compile with Pthread,*

```
$ g++ <file name> -o <output name> -pthread
```

select()

- *Select() provides you to supervise multiple sockets, telling you which is able to read or write, etc.*
- *With select(), it is possible to achieve Asynchronous Blocking I/O.*
- *For more details about I/O, you can go to this website.* **(Report 4)**
- *If you want to implement this assignment with select(), please refer to this website.*

select()

- *Monitor whether there is at least one fd available*

```
#include <unistd.h>
int select(int nfds, fd_set*, readfds, fd_set* writefds, fd_set* exceptfds,
struct timeval* timeout);     //return : 1 if it's success, -1 otherwise
```

- *nfds: specifies number of file descriptors to monitor.*
- *readfds: specifies the pointer to read file descriptor list*
- *writefds: specifies the pointer to write file descriptor list*
- *exceptfds: specifies the pointer to error file descriptor list*
- *timeout: deadline for select().*

select()

```
void FD_SET(int fd, fd_set *set);
void FD_CLR(int fd, fd_set *set);
int FD_ISSET(int fd, fd_set *set);  // return: 1 if it's available, else: 0
void FD_ZERO(fd_set *set);
```

- *FD_SET: Add the file descriptor into the set*
- *FD_CLR: Remove the file descriptor from the set*
- *FD_ISSET: Check if the file descriptor is available*
- *FD_ZERO: Clear the set*

OpenCV

- *Is an opensource library for computer vision.*
- *Mat is an image container to load an image so that you can easy to do image processing, recognition, etc.*
- *In this assignment, we use this library to get frames from videos on server, and show frames on client.*
- *An example code in openCV.cpp. tmp.mpg is an .mpg video file if you need.*
- *To compile code with OpenCV,*

```
$ g++ <file name> -o <output name> \
  `pkg-config –cflags –libs opencv`
```

FFmpeg

- *Is an opensource library for encoding / decoding video and radio.*
- *AVPacket is the slices of data, so that we can use AVCodec to decode AVPacket into frame.*
- *In this assignment, we use this library to get AVPacket and AVCodec from videos on server, and Using them to decode into frames and showing on client.*
- *An example code in ffmpeg.c, ffmpeg_install.sh is a script to install ffmpeg lib.*

Installation

```
$ bash ffmpeg_install.sh
```

```
ffmpegExample/
├── ffmpegExample.c
├── ffmpeg_install.sh
└── ffmpeg/
        └── ffmpeg3.3/
                ├── ffmpeg-lib/
                └── many many other files
```

Compile

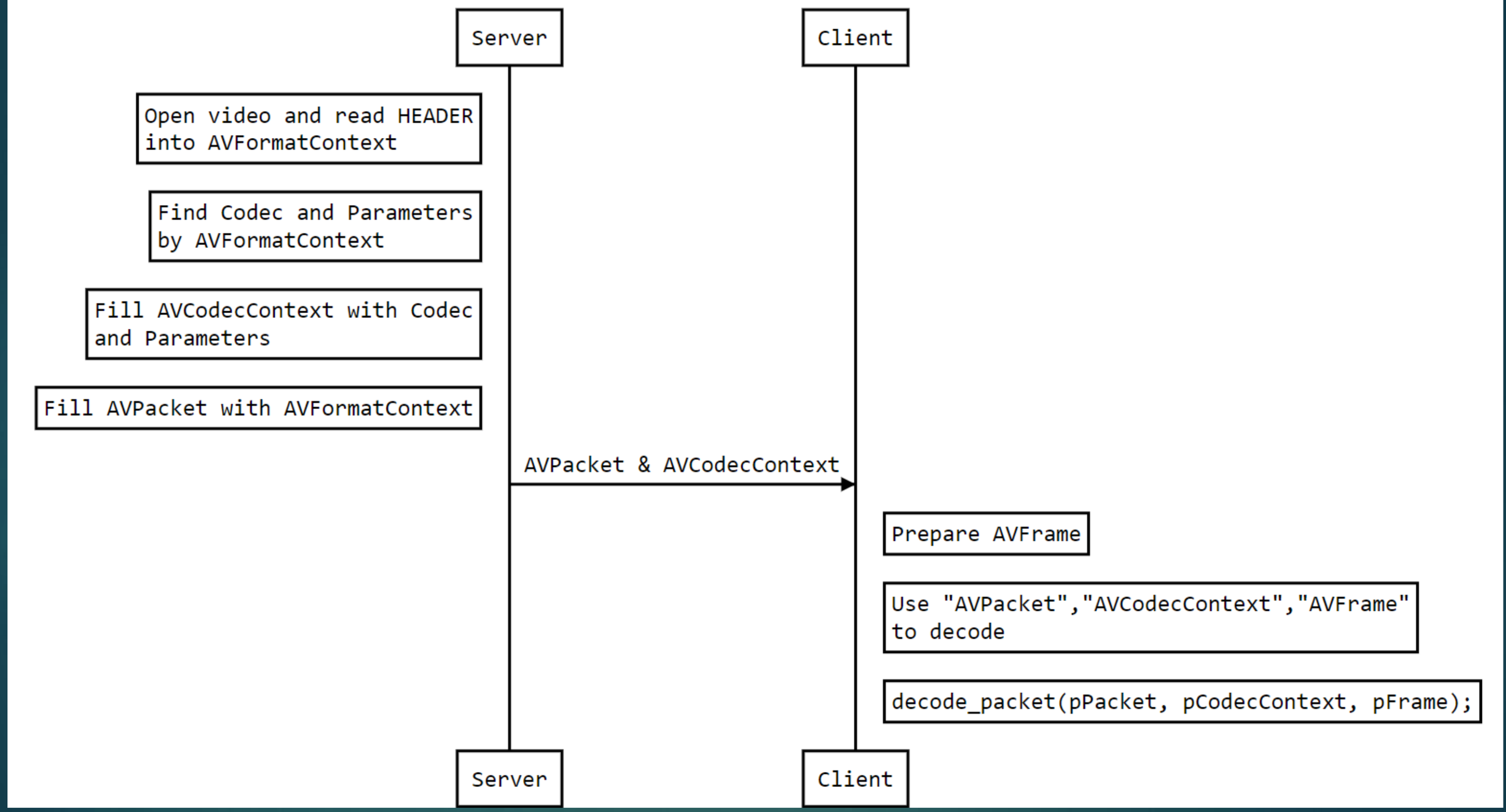
// please copy ffmpeg-lib/ folder to ffmpegExample/

// like this:

```
ffmpegExample/
    ├── ffmpegExample.c
    └── ffmpeg-lib/
```

```
$ export LD_LIBRARY_PATH=ffmpeg-lib/lib
$ gcc -I ffmpeg-lib/inc ffmpegExample.c -L ffmpeg-lib/lib -lavdevice -lavfilter -lavformat -lavcodec -lswresample -lswscale -lavutil -o ffmpegExample.out
```

FFmpeg


Server
Client
Open video and read HEADER into AVFormatContext
Find Codec and Parameters by AVFormatContext
Fill AVCodecContext with Codec and Parameters
Fill AVPacket with AVFormatContext
AVPacket & AVCodecContext
Prepare AVFrame
Use "AVPacket","AVCodecContext","AVFrame" to decode
decode_packet(pPacket, pCodecContext, pFrame);
Server
Client

Reference

- *Beej's Guide to Network Programming (中文)*
- *Beej's Guide to Network Programming (English)*