CS421 - Computer Networks Programming Assignment 2

ParallelFileDownloader

Due: 26 December 2021 at 23:59

In this programming assignment, you are asked to implement a program in either Java or Python. This program is supposed to download an index file to obtain a list of text file URLs and download these files **in parallel**. You are expected to use **multitheading** in order to be able to download parts of a file in parallel. You can find resources in the Internet about how to implement multithreading in Python and Java. For this programming assignment, you are not allowed to use any third party HTTP client libraries, or the HTTP specific core or non-core APIs. The goal of this assignment is to make you familiar with the internals of the HTTP protocol and using any (third party, core or non-core) API providing any level of abstraction specific to the HTTP protocol is not allowed. You must implement your program using either the Java Socket API of the JDK or the Socket package in your default Python distribution. If you have any doubts about what to use or not to use, please contact us. For Python, the use of any class/function from http or the requests package is prohibited.

Your program must be a console application (no graphical user interface (GUI) is required) and should be named as *ParallelFileDownloader* (i.e., the name of the class that includes the main method should be *ParallelFileDownloader* for Java submissions). If you are using Java, your code should run with the following command after it is compiled:

```
java ParallelFileDownloader <index_file> <connection_count>
```

If you are using Python, it should run with:

```
python3 ParallelFileDownloader.py <index_file> <connection_count>
```

command, where `<index_file>` is the URL of the index that includes a list of text file URLs to be downloaded and `<connection_count>` is the number of connections to be established for each file URL.

When a user enters the command above, your program will send an HTTP GET request to the server in order to download the index file with URL `<index_file>`. If the index file is not found, the response is a message other than 200 OK. In this case, your program will

print an error message to the command-line and exits. If the index file is found, the response is a 200 OK message. When this is the case, your program will print the number of file URLs in the index file and send an HTTP HEAD request for each file URL in the index file.

Requested file is not found If the requested file is not found in the server, the response is a message other than 200 OK. In this case, your program will print a message to the command-line indicating that the file is not found.

Requested file is found If the requested file is found in the server, the response is a 200 OK message. When this is the case, your program establishes `<connection_count>` **parallel connections** with the server including the file, downloads non-overlapping parts of the file through these connections, constructs and saves the file under the directory in which your program runs. The name of the saved file should be the same as the downloaded file and a message indicating that the file is successfully downloaded is printed to the command-line.

With your code, you need to submit a brief report describing how your code works.

Assumptions and hints

- Please refer to W3Cs RFC 2616 for details of the HTTP messages.
- You will assume that each line of the index file includes one file URL.
- You will assume that the name of each file in the index is unique.
- Your program **will not save the index file** to the local folder.
- The number of bytes downloaded through each connection should differ by at most one byte. The number of bytes downloaded through each connection is n/k if n is divisible by k, where n and k respectively denote the number of bytes in the file and the number of connections. Otherwise, $(\lfloor n/k \rfloor+1)$ bytes should be downloaded through the first $(n-\lfloor n/k \rfloor * k)$ connections and $\lfloor n/k \rfloor$ bytes should be downloaded through the remaining connections.
- Your program should print a message to the command-line to inform the user about the status of the files.

- The downloaded file should be saved under the directory containing the source file *ParallelFileDownloader.java or ParallelFileDownloader.py* and the name of the saved file should be the same as the name of the downloaded file.

  You may use the following URLs to test your program:
  www.cs.bilkent.edu.tr/~cs421/fall21/project1/index1.txt
  www.cs.bilkent.edu.tr/~cs421/fall21/project1/index2.txt

  Please contact your assistant if you have any doubt about the assignment.

Example

Let `www.foo.com/abc/index.txt` be the URL of the file to be downloaded whose content is given as

www.cs.bilkent.edu.tr/file.txt
www.cs.bilkent.edu.tr/folder2/temp.txt
wordpress.org/plugins/about/readme.txt
humanstxt.org/humans.txt

where the first file does not exist in the server and the sizes of the other files are 6000, 4567, and 1589 bytes, respectively.

Example run 1 Let your program start with the

```
java ParallelFileDownloader www.foo.com/abc/index.txt 3
```

or

```
python3 ParallelFileDownloader.py www.foo.com/abc/index.txt 3
```

command. Then all files except the first one in the index file are downloaded. After the connection is terminated, the command-line of the client may be as follows:

Command-line:
URL of the index file: www.foo.com/abc/index.txt
Number of parallel connections: 3
Index file is downloaded
There are 4 files in the index
1. www.cs.bilkent.edu.tr/file.txt is not found
2. www.cs.bilkent.edu.tr/folder2/temp.txt (size = 6000) is downloaded
File parts: 0:1999(2000), 2000:3999(2000), 4000:5999(2000)
3. wordpress.org/plugins/about/readme.txt (size = 4567) is downloaded
File parts: 0:1522(1523), 1523:3044(1522), 3045:4566(1522)
4. humanstxt.org/humans.txt (size = 1589) is downloaded
File parts: 0:529(530), 530:1059(530), 1060:1588(529)

Example run 2 Let your program start with the

```
java ParallelFileDownloader www.foo.com/abc/index.txt 5
```

or

```
python3 ParallelFileDownloader.py www.foo.com/abc/index.txt 5
```

command. Then all files except the first one in the index file are downloaded. After the connection is terminated, the command-line of the client may be as follows:

Command-line:
URL of the index file: www.foo.com/abc/index.txt
Number of parallel connections: 5
Index file is downloaded
There are 4 files in the index
1. www.cs.bilkent.edu.tr/file.txt is not found
2. www.cs.bilkent.edu.tr/folder2/temp.txt (size = 6000) is downloaded
File parts: 0:1199(1200), 1200:2399(1200), 2400:3599(1200), 3600:4799(1200), 4800:5999(1200)
3. wordpress.org/plugins/about/readme.txt (size = 4567) is downloaded
File parts: 0:913(914), 914:1827(914), 1828:2740(913), 2741:3653(913), 3654:4566(913)
4. humanstxt.org/humans.txt (size = 1589) is downloaded
File parts: 0:318(318), 318:635(318), 636:953(318), 954:1271(318), 1272:1588(317)

Submission rules

You need to apply all of the following rules in your submission. **You will lose points if you do not obey the submission rules below or your program does not run as described in the assignment above**.

- The assignment must be submitted to Moodle**.** Any other methods (Email/Disk/ CD/DVD/Cloud Drive) of submission will not be accepted.
- Zip all of the downloaded files, your report in PDF format, and your source code for submission. The submission should only include a single **ZIP** file. Any other compression is not accepted.
- The name of the zip file must be **AliVelioglu20141222** if your name and ID are **Ali Velioglu and 20141222**, respectively. If you are submitting an assignment done by two students, the file name should include the names and IDs of both group members like **AliVelioglu20141222AyseFatmaoglu20255666** if group members are Ali Velioglu and Ayse Fatmaoglu with IDs 20141222 and 20255666, respectively. • For group submissions, **ONLY ONE MEMBER** must make the submission. The other member must **NOT** make a submission. If you want to team up with someone else, make sure that you are both in the **SAME** section.

- All the files must be in the root of the zip file; directory structures are not allowed. Please note that this also disallows organizing your code into packages. The archive should not contain any file other than the source code with .java or .py extension.
- The archive should **not** contain:

  – Any class files or other executables,
  – Any third party library archives (i.e. jar files),
  – Any text files,
  – Project files used by IDEs (e.g., JCreator, JBuilder, SunOne, Eclipse, Idea, or NetBeans etc.). You may, and are encouraged to, use these programs while developing, but the end result must be a clean, IDE-independent program.

- The standard rules for plagiarism and academic honesty apply; if in doubt refer to Academic Integrity Guidelines for Students and Academic Integrity, Plagiarism & Cheating.

Important Notes

General Submission Rules:

1-) Your submission should include source code(s) (.java/.py).
2-) The format of the report should be **PDF**. (Do not upload .doc, .docx or any other types).
3-) If you need to tell a specific issue about your code, you can also include one README file.
4-) Your submission should not contain any other files other than the source code(s) (.java/.py), report (.pdf) and optional README. No .txt files, no folders, no IDE related files should be included.
5-) Compress these files with **.zip** format. (.rar, .7z or any other compressing types will not be accepted.)
6-) Make sure to follow rules in the "Submission Rules" section like name of the zip file, method of the submission etc.
7-) **Do not** save index files.

For Python Submissions:

1-) The code should run with the "python3 ParallelFileDownloader.py <index_file> <connection_number>" command.

2-) Python version should be **3.6 or higher**. Other versions (like Python 2) are **not accepted**.

For Java Submissions:
1-) The code should run with the following commands:
Compile: "javac *.java"
Run: "java ParallelFileDownloader <index_file> <connection_number>"
2-) Java version should be **8 or higher.**
3-) The JDK should be Oracle JDK (**not** OpenJDK).

Operating System:
1-) Your code will be tested on Ubuntu 18.04 LTS. Make sure that your code works correctly in this operating system.

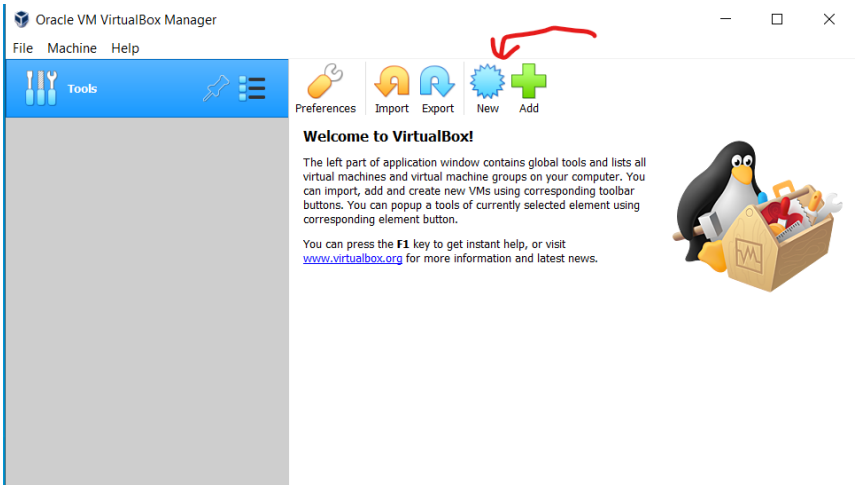
If you are using any other operating system, you can use a virtualization software to test your program. See "Virtual Box Guidance" section for further details.

PDF Content:
1-) Write a report explaining your code. No cover page is needed. It should **not** be more than 2 pages. We should be able to navigate the source code just from the report.

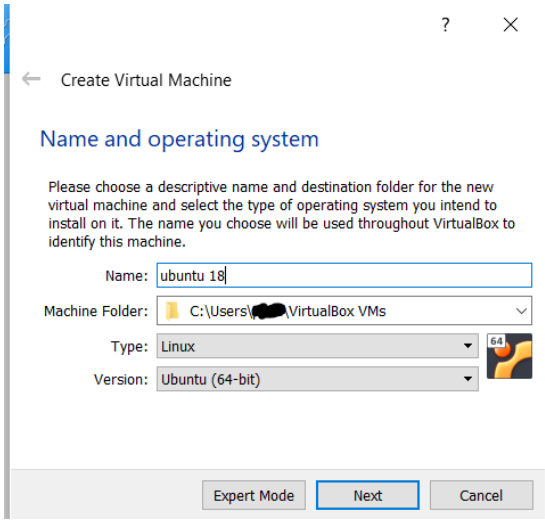
Virtual Box Guidance

If you are not using Ubuntu 18, you can use VirtualBox software. This software enables you to virtualize Ubuntu without actually installing it to your hard-drive.

If you want to use this software:
First, download it from "https://www.virtualbox.org/wiki/Downloads".
Secondly, download Ubuntu 18 from: "https://releases.ubuntu.com/18.04/".
Open the Virtual Box software and click "New":

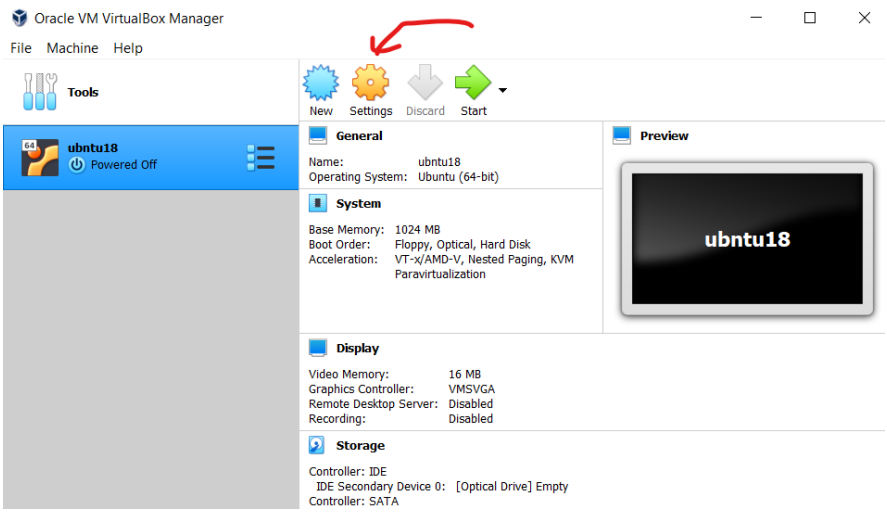


Give a name to your machine. Enter the corret type and version:

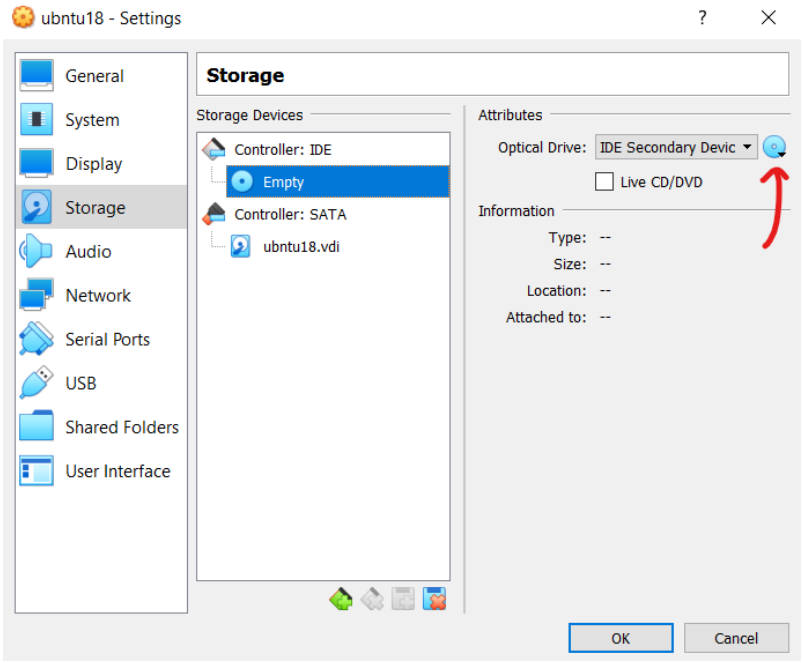


Adjust the rest of the settings as you wish. (You can use the default settings.).

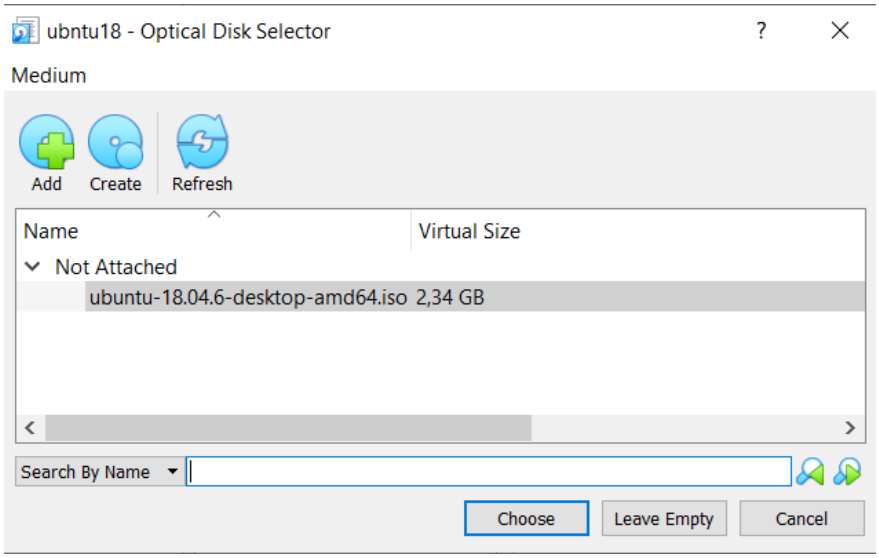
After you are done, click “Settings”:



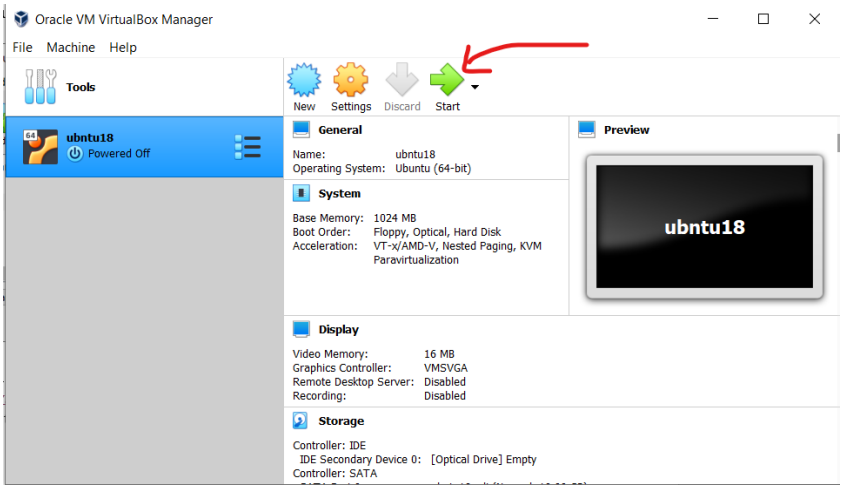
From the “Storage” tab, select the Ubuntu 18 image you downloaded by clicking on the highlighted icon and selecting “Choose/Create a Virtual Optical Disk”:



Find your Ubuntu image by clicking “Add”:



You can start using your virtual machine by clicking “Start”:



This tutorial (https://www.youtube.com/watch?v=QbmRXJJKsvs) also gives a good explanation in case you need more help. The steps are similar for all operating systems.