Server Packages:

Packages	Dependencies
Timer	None
Resources	None
Error	Resources
Get	Resources, Error
Put	Resources, Timer, Error
Core	Get, Put, Resources
Main.py	Core

Client Packages:

Packages	Dependencies
Timer	None
Resources	None
Error	Resources
Get	Resources, Error
Put	Resources, Timer, Error
Core	Get, Put, Resources
Request	Core
Main.py	Request

Timer package:

This package will be use by the sender part of the server and the client.
It will only used to start, restart and cancel timer when the one of those system send data to the other.

Content:

```
StartTimer()
CancelTimer()
RestartTimer()
```

Resources package:

This package will contain some defined resources for the core package and the main file. It will be used by Get, Put and Core Package on both system.
Content:

```
DATA_SIZE = 512
READ_REQUEST = 0x01
WRITE_REQUEST = 0x02
DATA = 0x03
ACK = 0x04
ERROR = 0x05
OCTAL_MODE = "octet"
MAX_TIMER_EXCEED = 6
TIMER = 10
```

Error Package:

Both will send error to the other part but works differently on the client and server side.

- On client side:

This package is used to print understandable error for the user while he try to send a file to the server or when he retrieve a file from the server. It won't send error when try to sending a file that the user can't access, this case is solved by the Request package, it is ask on Request package if you would overwrite the local file with the distant file. Error message send by this package will be:

➔ not defined (0)
➔ disk full or allocation exceeded (3)
➔ unknown transfer id (4)

Content:

```
__error_not__defined__ = 0x00
__file_not_found__ = 0x01
__access_violation__ = 0x02
__disk_full__ = 0x03
__id_error__ = 0x04
__illegal_operation__ = 0x05
__file_exist__ == 0x06

SendError()
IsCritical()
PrintError()
```

- On server side:

This package is used to get error from the client while the client is transferring a file and exit the communication between them. It used also to send error message to the receiver. Error message send by this package will be:

- ➔ Untitled 1not defined (0)
- ➔ file not found (1)
- ➔ access violation (2)
- ➔ disk full or allocation exceeded (3)
- ➔ unknown transfer id (4)
- ➔ illegal tftp operation (5)

Yes on our tftp server we will allow rewrite files, it's why we didn't use error n°6 which mean file already exist.

Content:

```
__error_not__defined__ = 0x00
__file_not_found__ = 0x01
__access_violation__ = 0x02
__disk_full__ = 0x03
__id_error__ = 0x04
__illegal_operation__ = 0x05
__file_exist__ == 0x06

SendError()
IsCritical()
```

Put package:

This package is used to send file to a receiver, no timer is requested for this part.
It will be used by the Core package and need Resources package.
On the server side it require Error package for sending error to client (see server Error package).

Content:

```
SendFile()
SendAck()
```

- On server side:

SendFile() may use SendError() with error code: 0, 1, 2, 4.
SendAck() may use SendError() with error code: 0, 3, 4.

- On client side:

SendFile() may use SendError() with error code: 0, 4.
SendAck() may use SendError() with error code: 0, 3, 4.

Get package:

This package is used to retrieve data/ack form the sender/receiver.
It will be used by Core package and need Resources, timer package.
On the client side it require Error package to print understandable error (see client Error package description).

Content:

```
GetAck()
GetData()
```

- On client side:

getAck() must call IsCritical() on receiving error instead of ack message
getData() must call IsCritical() on receiving error instead of data message

If the error is critical the function IsCritical() will return -1 and then we will print the error message, close the connection and exit program.

- On server side:

getAck() must call IsCritical() on receiving error instead of ack message
getData() must call IsCritical() on receiving error instead of data message

If the error is critical the function will IsCritical() return -1, closing the connection and then we will back to wait for connection loop.

Core package:

This package is the application core system. It's where we determine which request is used.
It will open file to read/write data.
This package depends on Put, Get and resources.

Content:

```
GetAnswers()
```

Request package:

This package is used over the Core package on the client side for 2 reasons, first it will determine
file access error, then it will use the Core package in the same way as the Main.py file of the server.

Content:

RequestConnection()

Main.py:

This file will be the main file to run while running the client/server.
There working differently on both side.

- On server side:

On server side the main program require one option:

--p/-port : port number to use

Content:

WaitForConnection()

- On client side:

On client side the main program require at least three options:

--H/-hostname	:	server fqdn or ip address
--p/-port	:	port number to connect to the server
-m/-mode	:	(optional) allows read/write default will be read
-i/-input	:	input file
--o/-output	:	(optional) output file default will be the input file
--h/--help	:	displaying usages

Content:

main()