# Running commands on other computers and transferring files between computers

1

# Remote Login

- Login to remote computer and run programs on that computer
  - Once logged in to remote computer, everything you type is executed on that remote computer
- Old, insecure protocol: `telnet`
- Secure protocol: `ssh` (Secure Shell)
  - `ssh ADDRESS`
  - `ssh -l USERNAME ADDRESS`
  - `ssh ADDRESS 'COMMAND'`

2

Normally you login to a computer that you have direct physical access to, i.e. the computer you are sitting at. However you can also login to another computer across the network, i.e. a remote login. Once logged in to the remote computer, everything that you type is executed on that remote computer, and the results are shown on your screen. It is as if you are sitting at that remote computer.

In the past, a protocol for remote login was called `telnet`. However that is not used very often any more because it is in secure (your username and password, as well as everything you did on the remote computer, are easy to be seen by others in the network).

The most common protocol used for remote login is ssh or Secure Shell. As the name suggests, your login credentials and everything you do on the remote computer is secured across the network using encryption. There are different implementations of SSH clients and servers. A common client/server is called OpenSSH. It is available on most Linux/BSD distributions. A common client for Windows is PuTTY.

Assuming there is a SSH server on the remote computer (OpenSSH server is already installed on the lab computers), to login you use ssh on the command line. There are many different options, three common ones are shown on the slide:

• You must provide the ADDRESS of the remote computer, e.g. IP or hostname

• You may optionally give the USERNAME that you want to log in to with on the remote computer

• Normally when using SSH you interactively enter commands on that remote computer. But if you want to run just one command on that remote computer, you can pass the COMMAND as a parameter after the address.

## Simple File Transfer

- Netcat (`nc`): opens TCP connection as client or server
- Example – transfer file.txt to 10.10.6.210
  - On server:

    ```
    nc -l 12345 > file.txt
    ```

  - On client:

    ```
    cat file.txt | nc 10.10.6.210 12345
    ```

  - Optional: time the transfer

    ```
    time cat file.txt | nc 10.10.6.210 12345
    ```

3

There are different ways to transfer a file between one computer and another. Here we look at a very simple method, using the program netcat or `nc`.

Netcat is a program that can opens TCP connections as either client or server. Once a TCP connection is established, you can send data through it. Once way to use netcat to transfer a file is to use it on both client and server. Lets say we have a file called `file.txt` on the client computer and want to transfer it to the computer 10.10.6.210 (the server computer with respect to netcat).

First, use netcat in server mode, starting it and listen (`-l`) on some port number. In this example port 12345 is chosen, but generally you can choose any unused port number (if you have permission to). Anything sent by a TCP client to port 12345 will be displayed by netcat on the screen. But we want it saved in a file. Hence we redirect the output of netcat to a file using `> file.txt`. Now anything received on port 12345 will be saved in `file.txt`.

Now on the client we will use netcat to open a TCP connection to the server. Normally the netcat client will take whatever you type and send it across the connection. Instead we want to take the contents of `file.txt` and sent that across the connection. Therefore display (cat) the file, and pipe (|) the contents into the netcat client, which in turn sends it across the TCP connection to the server (which saves what it receives in a file).

Often we'd like to measure how long a command takes. In this case, if we measure how long it takes to cat the file across the TCP connection, we can calculated the throughput (file size divided by time taken). You can precede any Linux command with `time` and it will run the command as normal, and then report the time it took to execute the command. Focus on the real time.


SSH also allows you to transfer files between computers. It has its own command for this called `scp`, for secure copy. It copies a file from some source to some destination, where normally either the source or destination is a remote computer. As the name suggests, this is done securely (encryption is used).

The general format for scp is:

```
scp SOURCE DESTINATION
```

Where SOURCE and/or DESTINATION is a file or path (meaning a file on your computer) and can include a URL that indicates the address of the remote computer.

The first example on the slide shows copying file.txt from your computer to the destination computer 10.10.6.210 and putting it into the directory /home/student/

The second example on the slide shows copying a myfile.txt from the directory /home/student on the remote computer 10.10.6.210 and putting it into the current directory on your computer.

There are many other options for `scp`, `nc` and `ssh`. Read the man pages to see them.

## Web Browsing on the Command Line

- Command line web page downloader

```
wget URL
```

  - e.g wget http://ict.siit.tu.ac.th/index.html downloads and saves index.html to my computer
  - Reports the time to download
- Use as simple throughput measurement
  - Download a large file and see the throughput

5

Assuming there is a web server on the destination computer, you can of course use your web browser (e.g. Firefox) to download files from that destination computer (as long as those files are in the web server directory, e.g. /var/www/). But what about using the command line to download files from a web server? There is a text-based web browser called lynx. Of course many graphical features of a website cannot be displayed. If you want to simply download the file (without viewing the marked up contents) you can use wget.

wget takes the destination URL as input, and downloads the file from the web server, saving the file on your computer. It also reports the status and summary of the download, including the time it took and rate (or throughput).

When to use wget? It is useful when you want to download files, but not view them immediately and when you want to automate the download of multiple files (e.g. download 100 images at some base URL).

You can also use wget as a simple throughput (performance) measuring tool. Download a large file from a server and the throughput/time is reported by wget, giving you the throughput for the path between browser and server.

wget has many command line options. Read the man page.

## Performance Testing

- iperf: Test throughput of link or network
- On server computer:
  - `iperf -s`
- On client:
  - `iperf -c SERVERADDRESS`
- Many options:
  - `-t`: time of test
  - `-u`: UDP test
  - `-b`: sending rate with UDP test

6

We often want to test the performance of a link or path, especially measure the throughput when transferring data. We've seen nc (combined with time) and wget can be used to measure throughput by transferring a file and timing how long it takes.

An application dedicated to measuring throughput is called iperf. To work, it must be run on both computers, server and destination.

First run iperf on the server computer (normally any computer can be an iperf server) using the -s option. Once you start it, iperf waits for an iperf client to connect.

Now run iperf on the client computer using the -c option and indicating the IP address of the server computer. Then iperf starts a performance test, by default sending data for 10 seconds using TCP as fast as possible. At the end, both client and server will report the performance of the data transfer (they should be about the same; if not, use the report from the server).

Iperf has many options to change how the test is performed. You can change the duration of the test (default: 10 seconds), the transport protocol used (default: TCP), and others. See the man page for iperf.