Faculty of Computer Science, Dalhousie University *10-Jan-2025*

DGIN 5201 — Digital Transformation

Lecture 3: Lab 1: Simple Web Experiments

Location: Goldberg CS 134 and 143 Instructor: Tymon Wranik-Lohrenz
Time: 11:35–12:25 and 13:05–13:55

Lab 1: Simple Web Experiments

Hands-on e1: Web Site, Shell, Permissions

Using `timberlea` Server

- `ssh` login into `timberlea.cs.dal.ca`
- Windows: you can use the program PuTTY
  - other options available; e.g., MobaXterm
- On Mac: open a Terminal and type:
  ```
  ssh <your_csid>@timberlea.cs.dal.ca
  ```
  where instead of `<your_csid>` you should use your own CSID
- On Linux: similarly to Mac, you open the terminal and type the same command:
  ```
  ssh <your_csid>@timberlea.cs.dal.ca
  ```

You can now try to login to the `timberlea` server provided by the Dal FCS computing environment. Your own computer may be a Windows machine, a Mac, or a Linux. You need to use the `ssh` secure protocol to login to `timberlea` and on each of these environment you may need a different application to login.

On Windows: If you use a Windows environment, you can use a well-known open-source and free program named PuTTY to login. This will be explained in the next step. If you do not have the PuTTY program, you can install it from the Internet.

To download PuTTY from Internet, you can search it using Google. However, you should not download just any copy of it for security reasons. The official site of PuTTY is:

`http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html`

Putty is a free implementation of Telnet and SSH for Windows and Unix, along with an xterm terminal emulator.

On Mac: In Mac OS environment, you can click on the search image in the upper right corner and type ‘Terminal’ to find the `Terminal` application. Once you open the terminal, you can login to the `timberlea` server by typing:

```
ssh <your_csid>@timberlea.cs.dal.ca
```

where `<your_csid>` is your CSID userid.

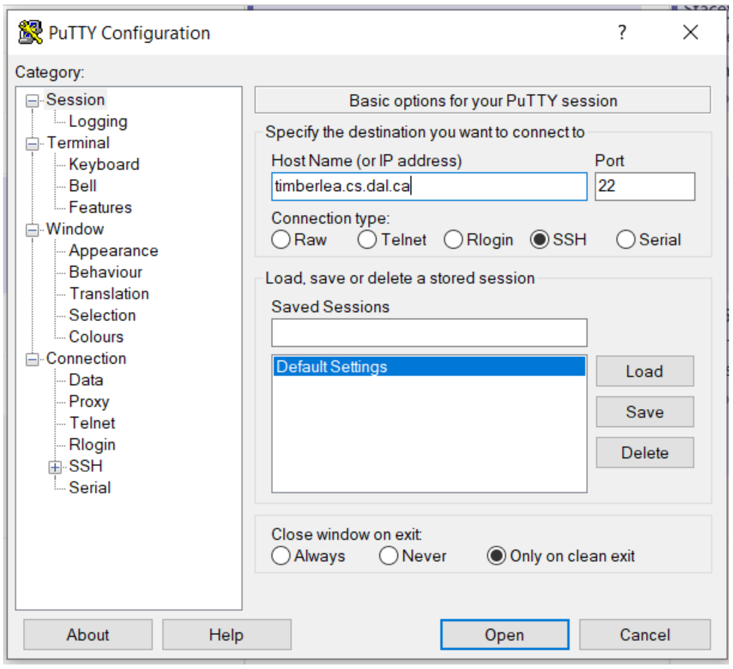
On Linux: In a Linux environment, you should, similarly to Mac, open a terminal and type:

```
ssh <your_csid>@timberlea.cs.dal.ca
```

where `<your_csid>` is your CSID userid.

Running PuTTY

– Double-click the PuTTY icon, and the following window should appear:

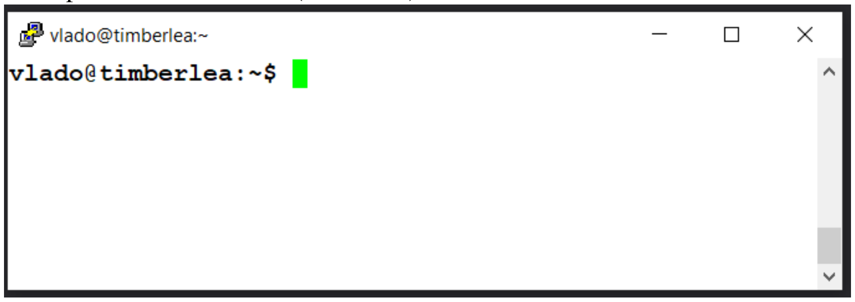


Specify the destination you want to connect to
Host Name (or IP address): timberlea.cs.dal.ca
Port: 22
Connection type: Raw Telnet Rlogin [✓ SSH] Serial
Load, save or delete a stored session
Close window on exit: Always Never [✓ Only on clean exit]

You should fill in the basic information: `timberlea.cs.dal.ca` for the Host Name. Make sure that the port number is 22; i.e., Connection type is SSH. You click 'Open' and the login process should start. You are likely to receive a warning about an unknown host key. Normally, this is something that you should be careful about and try to make sure that the offered fingerprint matches the fingerprint of the server, but in a relatively secure network you can accept this connection. Once accepted, the host key is stored with PuTTY and this warning should not appear again.

Hands-on Exercises

– You should use PuTTY or another client to login to `timberlea`
– FileZilla is a good tool to copy files back and forth, but does not provide access to command-line (shell)
– The following exercises should be finished and will be graded as a part of Assignment 1
– Example of command-line (bash shell) access:



```
vlado@timberlea:~$
```

Creating a Simple Web Page

– Try command: `pwd`

- Enter directory: `public_html`
- Create directories: `dgin5201/e1`
- Set permissions for this directory to be all-accessible: `chmod` command
- Go to directory `dgin5201/e1` and create file `index.html` with the following content:

```
<html><body>
This is a very small HTML file.
</body></html>
```

- Make `index.html` all-readable and access it over Web

Opening Web Page in a Browser. Now, we can try to open our web page by entering the following URL in a web browser:
`http://web.cs.dal.ca/~`*userid*`/dgin5201/e1`
As usual, the `userid` should be replaced with your CS userid. The browser should show our very simple web page.

Concepts Review: Example 1

- `ssh` access, PuTTY, `bash` shell
- `bash` commands: `pwd, ls, cd, mkdir, chmod, rmdir`
- File permissions
- Text editors: `emacs, vi, pico, nano`, or use remote editing: FileZilla, `vscode`
- Emacs editor:
  `emacs index.html` or `emacs -nw index.html`
  `C-x C-s` to save, `C-x C-c` to exit, `C-h t` to go through simple tutorial (`C-` means $\boxed{\text{Ctrl}}$ and other key)
- HTML: simple tags, `html, body`
- Web and HTTP access

Requirements of `e1`

- At the end of Example 1 (e1), there should be the following directories (folders), files and their permissions:

```
~/public_html/dgin5201               rwx--x--x
~/public_html/dgin5201/e1            rwx--x--x
~/public_html/dgin5201/e1/index.html rwxr--r--
```

- Content of `index.html` was given previously

File Permissions Review

- Each file or directory has user owner and group owner (group of users)
- Permissions defined for: user, group, other
- Each of these have three permissions: `rwx` — read, write, execute
- For directories 'execute' means actually access
- Examples of using chmod: `chmod 664 file.txt`

```
chmod og-r file.txt
chmod u+x,og+r file.txt
chmod u=rw,og= file.txt
chmod a+r file.txt; chmod -R u+r+w+X dir1
```

Hands-on e2: Printable Page, Files Shared

Example e2: User Registration, Printable Page, Files Shared

- Consider a Conference Management System: *CoMS*
- Let us build a conference registration form
- We also want to provide them with some material
- First iteration: Create a printable form
- Create directory `public_html/dgin5201/e2`
- Go to that directory
- Add file `index.html` (content to be given)
- Make sure that the permissions of `e2` are `rwx--x--x`, and of `index.html` are `rwxr--r--`

Example 2: `public_html/dgin5201/e2/index.html`

```
<html><head><title>Conference Registration</title></head>
<body>
<h1>Conference Registration</h1>

<p>This is a registration page for CoMS.<br/>
For additional documents, please check <a
href="material">here</a>.<br/>
Please enter your information below to register:

<table>
<tr><th align=right>First and last name:</th>
<td>________________________________</td></tr>
<tr><th align=right>Email:</th>
<td>________________________________</td></tr>
<tr><th>Area of Interest (DB, HI, DS):</td>
<td>________________________________</td></tr>
</table>
```

Example 2: Make material available

- Create readable and accessible ('executable') directory `material` (permissions: `rwxr-xr-x`)
- Copy PDF from: `~vlado/public/dt-mini-conf.pdf` into directory `material`
- Setup permissions for the directory `material` to be all readable and accessible (`rwxr-xr-x`), and for the file `dt-mini-conf.pdf` to be all readable (`rw-r--r--`)
- Try to access material link on the page. Does it work? Why?

Example 2: Prepare `.htaccess` in `material` directory

- Prepare file `.htaccess` and make it all readable (`rw-r--r--`):
  ```
  Options Indexes
  ```
- Check material access now
- Add the following line to `.htaccess` and try accessing again:
  ```
  Options Indexes
  AddDescription "DT Conference Poster (PDF)" dt-mini-conf.pdf
  ```
- Add "and Information" to "DT Conference Poster" and access

- Add the following line and try again:

```
Options Indexes
IndexOptions DescriptionWidth=*
AddDescription "DT Conference Poster..." dt-mini-conf.pdf
```

- `.htaccess` file is used to configure Apache web server behaviour
  - can be used to provide a simple password-protected access

Concepts Review: Example 2

- Creating something that looks like form when printed
- HTML tags: `head, title, h1, p, br, a, table, tr, th, td`
- HTML attribute: `<th align=right>` `<a href="...">`
- `bash` shell: `cp`, using path, `~vlado`
- Accessing directory via browser
- `.htaccess` file for the Apache server: `Options Indexes, AddDescription`

Hands-on e3: Password Protection

Example e3: Next Iteration of Our Site: Password Protection

- Let us make a copy of our `e2` site
- First, go back to the directory above `e2`:

```
cd ../..
```

- Use command `pwd` to check your directory
- Copy `e2` to `e3` as an exact copy:

```
rsync -av e2/ e3/
```

- Check the new site `e3` in the browser
- `rsync` is a very useful utility for copying directory structures
  - it works locally as well as over ssh
  - it copies incrementally differences, which is important if two sites are large and mostly equal
  - it may preserve permissions if we use option `-a`

Example 3: Simple Password Protection

- `cd` to `e3` directory and let us prepare a password
- In a locally-only readable file `pw` (`rw-------`) we can save a password for our reference: `dt dt5201`
- Prepare the password for the site using the command:

```
htpasswd -bc .htpasswd dt dt5201
```

- Make the file `.htpasswd` all-readable and check its contents
- Prepare the file `.htaccess` and make it all readable:

```
AuthType Basic
AuthName dgin5201
AuthUserFile /users/webhome/<your_csid>/dgin5201/e3/.htpasswd
AuthGroupFile /dev/null
<Limit GET POST>
require user dt
</Limit>
```

- Check that site is password-protected

Summary of e3

- Files and permissions copied from `e2`
- `pw` file with permissions `rw-------`
- `.htpasswd` file with permissions `rw-r--r--` and appropriate content set up with the `htpasswd` command
- `.htpasswd` file with permissions `rw-r--r--` and content set up for password protection as given in class

Concepts Review: Example 3

- `rsync` command, `-av` options
- `htpasswd` command, password saved as hash
- Using `.htaccess` for password-controlled access

Unix-style Customization

Slide notes:

Unix-style Customization

- Unix-style customization is typically text-based
- Example: bash customization
  - aliases: rm, mv, cp, em
  - .profile and .bashrc files
- Example: Emacs customization
  - .emacs file
- Earlier example: Apache customization
  - .htaccess, .htpasswd files

The Unix-style operating systems, such as Linux, mostly use text-based configuration in various scenarios, and it is useful to be able to edit those plain-text files to customize the system and different software utilities, and for solving different technical problems.

Bash shell customization: The Bash shell customization is one such example. For example, the command `rm` is used to remove a file. Its default behaviour is usually such that after a command such as `rm file1` it will immediatelly remove the file named `file1`. In practice, this may easily lead to a mistake of accidentally removing a file that we did not want to be removed. This can be fixed by creating an alias for the command `rm` in Bash using the command:

```
alias rm='rm -i'
```

This command would create the alias `rm`, which when typed into the shell causes the actual command '`rm -i`' to be executed, which uses the option `-i` of the program `rm` to always ask user for a confirmation if they really want to remove the given file.

Aside: Touch Typing

- If you don't use touch typing, consider learning it
- A relatively simple and not popular skill, but
  - actually important, and even more and more relevant

Hands-on e4: Introducing a Form

Example e4: Introducing a Form

- With `rsync` copy `e3` to `e4`, update .htaccess file
- Change the table part of `index.html` to:

```
<form>
<table>
<tr><th align=right>First and last name:</th>
<td><input type="text"></td></tr>
<tr><th align=right>Email:</th>
<td><input type="text"></td></tr>
<tr><th>Area of Interest (DB, HI, DS):</td>
<td><select><option>DB</option><option>HI</option>
  <option>DS</option></select></td></tr>
 </table>
 </form>
```

- Check the page and see that this is usable fillable form, which can be printed

Concepts Review: Example 4

- Creating fillable form in HTML: `<form>...<form>`
- `<input type="text">`
- `<select><option>op1</option>...</select>`

Summary of e4

- Files set up as in `e3`
- `index.html` modified to make a usable fillable form