VE482 — Introduction to Operating Systems

Lab 3

Manuel — UM-JI (Fall 2020)

Goals of the lab

- Be ready to use git in the course
- Work with source code
- Prepare for project 1
- Regular expressions

1 Project 1: presentations (part 2)

To ensure a more synthesized support during project 1, presentations are split into two parts. Topics are available on Canvas and their selection is on a first come first served basis.

Please well prepare your presentation and ask questions on others' research. This should greatly help in the development of your `mumsh`. Be careful, mum might be listening!

2 Git in VE482

Go to http://learngitbranching.js.org/ and complete the following levels:

- Main → Introduction Sequence: all;
- Main → Ramping Up: all;
- Main → A mixed bag: 1, 4;
- Remote → Push & Pull – Git Remotes!: 1-4, 6;

3 Working with source code

3.1 The `rsync` command

In Unix-like systems the `rsync` program allows to synchronise different folders on the same system or over the network. When applying some changes to the source code it is highly recommended to have a copy of the original version such as to be able to revert back to the previous version in case of problem.

Proceed with the following steps:

- In Minix 3 install the `rsync` software
- Install `rsync` on you Linux system
- Read `rsync` manpage
- Create an exact copy of the directory `/usr/src` into the directory `/usr/src_orig`
- If you have altered Minix 3 source code during homework 2 remove your changes from `/usr/src_orig`
- Create an exact copy of the Minix 3 directory `/usr/src_orig` into your Linux system, using `rsync` and `ssh` (note that the `ssh` server must be activated under Linux)

3.2 The `diff` and `patch` commands

When dealing with source code two main situations are likely to arise: (i) you want to share your changes with others, or (ii) you want to apply changed performed by someone else.

Most of the time updates on source code concern few lines scattered over several files. Therefore instead of sharing all the files it is much more convenient to only specify which lines should be updated, and how. This is the role of the `diff` command. The `patch` command is used to apply the changes previously created with `diff`. Both `diff` and `patch` programs should already be installed in your OS.

Proceed with the following steps:

- Read the manpages of `diff` and `patch`
- Using the `diff` command, create a patch corresponding to your changes in homework 2
- Retrieve your patch on your Linux system
- Apply your patch to the copy of `/usr/src_orig` on your Linux system
- Revert the patch

3.3 Remarks

The programs `rsync`, `patch` and `diff` are very useful however when big projects are managed by many people at the same time they are not convenient to handle. A more advanced, automatised approach is required such as to help solving collisions in a more simple way. For instance user *A* commits some changes on the initial version of the file `foo.c`. Then user *B* does the same. Notice that changes made by *B* may collide with updates from *A*. To prevent such issues *B* should have worked based on *A*'s version of the `foo.c` file.

To overcome such kind of issues and render things smoother and easier several systems were created; at the moment the most commonly used is called `git`, older ones such as `svn` or `cvs` are still used in some places.

In the remainder of this course you will be required to use the ve482 `git` server in order to keep track of your project work.

4 Scripting and regular expressions

Two programming languages often used in conjunction with Bash are `sed` and `awk`.

- Using `curl` or `wget` retrieve information on shanghai air quality and pipe it to `sed` which should parse the output in order to display the information in the terminal following the format below
  `AQ: value Temp: value ºC` (e.g. `AQ: 55 Temp: 24 ºC`).
- Pipelining the output of `ifconfig` to `awk` return only the ip address of your current active network connection (the active network interface can be passed to `ifconfig`).