Introduction to `awk` programming

Michael F. Herbst

15. – 17. August 2016

Course description and content

Dealing with large numbers of plain text files is quite frequent when making scientific calculations or simulations. For example, one wants to read a part of a file, do some processing on it and send the result off to another program for plotting. Often these tasks are very similar, but at the same time highly specific to the particular application or problem in mind, such that writing a single-use program in high-level language like `C++` or `Java` hardly ever makes much sense: The development time is just too high. On the other end of the scale are simple shell scripts. But with them sometimes even simple data manipulation becomes extremely complicated or the script simply does not scale up and takes forever to work on bigger data sets.

Data-driven languages like `awk` sit on a middle ground here: `awk` scripts are as easy to code as plain shell scripts, but are well-suited for processing textual data in all kinds of ways. One should note, however, that `awk` is not extremely general. Following the UNIX philosophy it can do only one thing, but this it can do right. To make proper use of `awk` one hence needs to consider it in the context of a UNIX-like operating system.

In the first part of the course we will thus start with revising some concepts, which are common to many UNIX programs and also prominent in `awk`, like *regular expressions*. Afterwards we will discuss the basic structure of `awk` scripts and core `awk` features like

- ways to define how data sits in the input file
- extracting and printing data
- control statements (`if`, `for`, `while`, . . .)
- `awk` functions
- `awk` arrays

If there is time left we will also look at some advanced topics, like performing calculations with arbitrary precision using `awk`.

This course is a subsidiary to the bash course which was offered in August 2015.[1]

[1] `http://blog.mfhs.eu/teaching/advanced-bash-scripting-2015/`

Learning targets and objectives

After the course you will be able to

- enumerate different ways to define the structure of an input file in `awk`,
- parse an structured input file and access individual values for post-processing,
- use regular expressions to search for text in a file,
- find and extract a well-defined part of a large file without relying on the exact position of this part,
- use `awk` to perform simple checks on text (like checking for repeated words) in less than 5 lines of code.

Prerequisites

- Familiarity with a UNIX-like operating system like GNU/Linux and the terminal is assumed.
- Basic knowledge of the UNIX command `grep` is assumed. You should for example know how to use `grep` to search for a word in a file.
- It is not assumed, but highly recommended, that participants have had previous experiences with programming or scripting in a UNIX-like operating system.