Using Shells and Scripting (slides)

Intro

In the intro portion of the workshop you will learn:

- What are shells?
- Which one(s) to use?
- What is scripting and what are its applications?
- How to write scripts.
- How to use some useful tools in scripts.

What are shells?

A shell is the program that is started after logging on a Linux machine and that allows users to type commands (aka the login shell). For historical reasons there are several shells:

1. The Bourne shell (`sh`) or the Bourne again shell (`bash`)
2. The C shell (`csh`) or the t C shell (`tcsh`)
3. The Korn shell (`ksh`)

You can check the Introduction to the Command Line for Genomics carpentries lesson for an intro to `bash`.

Everything about each shell is explain in the "man pages", there are books written about scripting and there is a ton of info on the web too.

Which shell(s) to use?

On Linux machines `sh` and `bash` are the same program, `csh` and `tcsh` are also the same program.

- `bash` is the most used shell by biologist (and to install packages for historical reasons)
- `csh` is the most used shell by astronomers. I was written after `sh` to have a more C like syntax (the C programming language, hence the name) and the `tcsh` is an improvement to the C shell for terminal use.
- a few people and systems use the Korn shell.

You can write scripts in any shell syntax you care, independently of what your login shell is.

What is scripting and what are its applications?

Scripting vs. Programming

- Programming languages are compiled to produce executables with machine level instructions ("complicated").
- By contrast, scripting refers to instructions that are parsed and executed by a program, hence ** there is no need to compile the script (convenient) ** but, *in principle*, they run more slowly than executables.
- Some scripting languages are parsed to check for syntax errors before executing them, but *not* shell scripts.
- Scripts can combine existing modules or components, while programming languages are used to build more sophisticated/complicated applications from scratch.
- Scripts are used to help run applications, on Hydra a job file is a (simple) script.
- You can write more complex scripts, and scripts can invoke

How to write scripts.

- A Script is a text file that holds a list of commands, and thus can be written with any type of editor (`nano`, `vi`, `emacs`).
- The commands in the script must follow the syntax of the shell used to parse the script
- A script can take arguments (options) and thus be more general, and allows you to define variables, use expressions or execute commands ** A variable is a way to hold a value and refer to it by its name, or a way to modify how some commands behave - variables can also be one dimensional arrays (lists) ** An expression allows to perform simple arithmetic or use commands to create values held by variables (for example: set the variable `num` to hold the number of lines in a file).
- The scripting syntax allows for "flow control" namely it allows for ** tests and logical operators - `if` statements ** loops - `for` statements ** more flow control: `case`, `while`, `until` and `select` ** bash also allows to define functions ** the

Script Variables

- A variable is a character string to which a value is assign.
- The assigned value can be a number, some text, a filename, device, or any other type of data.

Script Arguments

- Arguments are a way to supply parameters to a script.
- Arguments are useful when a script has to perform differently depending on the values of some input parameters.

Script Flow Control

Tests and Logical Operators**

- Logical operators can be used to test conditions and create complex expressions by combining conditions.
- These operators allow you to evaluate the truthfulness of a condition or multiple conditions, and provide a way to control the flow of execution of scripts

```
if   then
fi
```

Loops

- Loops allow us to repeat a command or set of commands for each item in a list.

Other Flow Control Instructions

Useful tools for scripting

Simple ones

- `sed` - the steam editor for filtering and transforming text
- `awk` - pattern scanning and processing language
- `tr` - translate or delete characters
- `cut` - remove sections from each line of files
- `bc` - An arbitrary precision calculator language
- `date` + -date=“specification” - date handling

More sophisticated ones

- `PERL` - practical extraction and report language
- `Python` - high level general purpose programming language
- `Tcl` - Tool Command Language
- etc...

scripting hands-on

In the hands-on portion of the workshop you will learn how to:

- Run a set of commands: Scripts
- Simplify and avoid errors
- Test assumptions
- Generalize a script to be used for multiple files or executions: Arguments
- Ease hands-on time by repeating a command for every file: for loops
- Re-use arguments by manipulating the text
- Get parameters from another file

Log in to Hydra

If you need a reminder about how to log into Hydra and how to change your password, check out our Intro to Hydra tutorial:
`https://github.com/SmithsonianWorkshops/Hydra-introduction/`